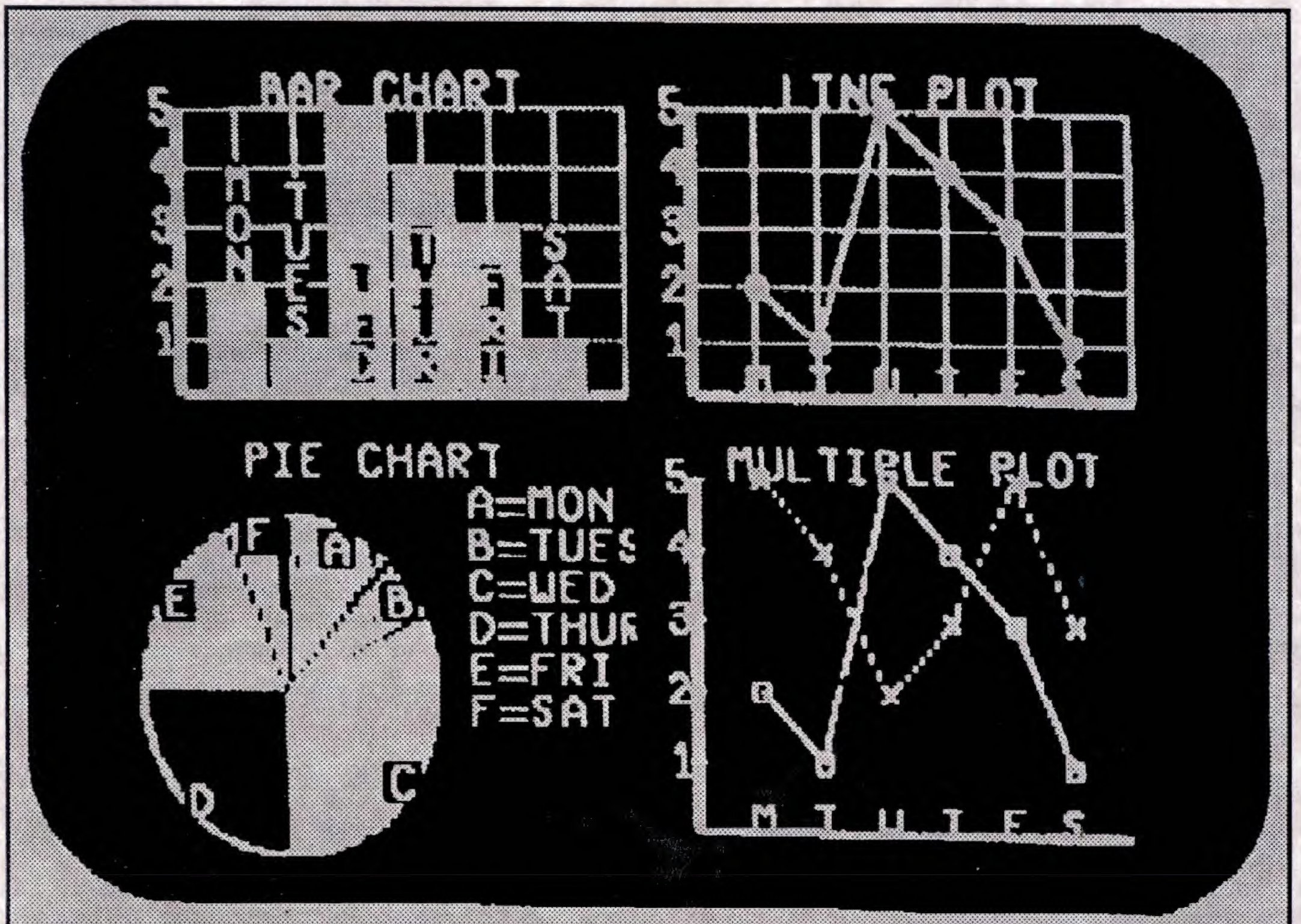
TRSTimes

Volume 3. No. 3. - May/Jun 1990 - $4.00



Type-in programs for Model I, III & 4, printer graphics, reviews, power programming, hints & tips, Assembly Language tutorial, humor & much more.

LITTLE ORPHAN EIGHTY

A funny thing happened on the way to the club. Well, actually I wasn't going to the club and, to be perfectly honest, it wasn't even funny. 'Strange' might describe the situation better.

I had just received my copy of LS-DOS 6.3.1., along with MISOSYS' editor/assembler package called EDAS.

Having a little breather between issues of TRSTimes, I sat down and began to play with my new DOS. First, of course, I backed up the original and put the master away. Then, having double-sided drives, I made a working double-sided boot disk. Going through the various DOS functions, I concluded that this upgrade ran smoother, and just a little faster, than its predecessor.

Next, I made a backup of the EDAS disk and began writing a program that Tim Sewell had requested some months ago. Around 3:30 in the morning I had finished a preliminary version, and I began to test it. Most of it worked as intended, with just a couple of cosmetic flaws - no problem. I would quit for the night and make the corrections the next day.

Since my program read and wrote to the directory, I decided to make a backup of my DOS disk -- you know, *just in case...*

I formatted a disk in drive :1 and backed up my double-sided DOS disk to it. The backup failed. 'Directory read error' was the message staring me in the face. 'Great programming, Lance', I disgustedly mumbled to myself, 'you just destroyed the directory!'

Enough for tonight, though. I turned off the computer, jumped in for a quick shower, and hit the bed. Fixing my 'great' code would have to wait until morning.

After a few hours sleep, another shower, and breakfast while reading the sports section of the LA Times, I fired up the Model 4 again. Rather than fix the code, I'd make a fresh double-sided DOS disk directly from the master. Having done that, the next thing was to backup the newly made DS DOS disk, as was my intention last night.

I formatted another DS disk in drive :1 and then started a mirror image backup. WHOA.... again it failed, and with the same error message - Directory read error. Hmmm!!

Well, maybe DISKCOPY would work. No such luck; it aborted at cylinder 20, reporting a directory error. Something was definitely wrong.

I called up Roy Beck, whom I knew had also just received LS-DOS 6.3.1, and asked him to create a double-sided DOS disk, and then make a mirror image backup of it. A few minutes later he called back and confirmed my findings. It wasn't my code at all - it was a problem with DOS.

Ok, it was time to find out just what was wrong so, since the errors reported pertained to the directory, I fired up UTILITY4 (a nice public domain ZAP program written by David Goben) and began reading the directory sectors one by one (cylinder 20). Everything was dandy until I reached sector 34. It, as well as sector 35, had not been initialized as directory sectors; instead they both contained the standard formatting pattern. Aha - now we were getting somewhere - the problem was obviously not in BACKUP/CMD nor in DISKCOPY/CMD. The villain had to be FORMAT/CMD.

Now, having written a couple of TRS-80 format routines in the past, I was somewhat familiar with what happens in FORMAT/CMD. Roughly speaking, it goes through a loop which formats each track, then goes through another loop which checks whether or not the track format was successful. Having done that, it begins writing system information (the little dots moving accross the screen), first the boot track (track 0), and then the directory track (track 20, if possible). Writing the directory sector is where the problem had to be. Yes, indeed!

Some years ago I disassembled FORMAT/CMD from TRSDOS 6.2.1 and commented a fair portion of the code. I found the folder where it was stored (amazing), and then began a screen disassembly of the LS-DOS 6.3.1 version of FORMAT/CMD. The question now was: "How different were they"? Luckily, they proved to be almost identical, at least the portion I was interested in.

To make a long story short, I eventually found the counter that checked the loop initializing the directory. Sure enough, it aborted after 34 loops, leaving the remaining two sectors uninitialized. Changing the counter to 36 solved the problem, and I now get backups and diskcopies of my double-sided disks without a hitch. The patch is:

PATCH FORMAT/CMD.UTILITY (D03,7E=24:F03,7E=22)

While I was at it, I went ahead and changed the DOS version from 6.3.1A to 6.3.1B. This patch is as follows:

PATCH BOOT/SYS.SYSTEM6 (D02,1F=42:F02,1F=41)

The author of LS-DOS 6.3.1 (Roy Soltoff) has been notified of the bug, and the above patches. I am now anxiously waiting to hear what the 'official' patches will be....stay tuned.

THE OFFICIAL PATCHES

Got the patches from Roy Soltoff just in time for this issue. It seems I was almost right. Rather than changing the counter, Roy elected to change the byte immediately following. As the reason is explained in detail in the Misosys Quarterly IV.ii, I will simply give the correct patches:

PATCH FORMAT/CMD.UTILITY (D03,7F=21:F03,7F=32)
PATCH BOOT/SYS.SYSTEM6 (D02,1F=42:F02,1F=41)

And now.....**Welcome to TRSTimes 3.3**

TRSTimes magazine

Volume 3. No. 3. - May/Jun 1990

PUBLISHER EDITOR
Lance Wolstrup
CONTRIBUTING EDITORS
Roy Beck
Dr. Allen Jacobs
TECHNICAL ASSISTANCE
Members of:
San Gabriel Tandy Users Group
Valley TRS-80 Users Group
Valley Hackers' TRS-80 Users Group

TRSTimes magazine is published bi-monthly by TRSTimes Publications. 20311 Sherman Way, suite 221. Canoga Park, CA. 91306. (818) 716-7154.
Publication months are January, March, May, July, September and November.

1990 subscription rates (6 issues):
UNITED STATES & CANADA:
$18.00 (U.S. currency)
ALL OTHER COUNTRIES:
$23.00 for surface mail, or
$29.00 for air mail.
(U.S. currency only)
Article submissions from our readers are welcomed and encouraged. Anything pertaining to the TRS-80 will be evaluated for possible publication. Please send hardcopy and, if at all possible, a disk with the material saved in ASCII format. Any disk format is acceptable, but please note on label which format is used. Also, please make sure that your name and address is written legibly on both hardcopy and disk label.

THE MAIL ROOM

CLAN

In issue 3.1, page 4, Robert Lutes of Mountain Home, AR expresses a preference for the CLAN geneology program, but states he has some uncorrected errors. I have several fixes from the author, before he went MS-DOS. I have used CLAN since 1986, and have solved most of the bugs, and I will gladly share what information I have. Recently I have been converting the DMP printouts for my RSDWP 220 printer, but still have a couple of glitches in that.

The new style and quality og TRSTimes are great. How about a couple of articles explaining how to use a Basic Compiler, such as the one 80 Micro published near the end of their Model 4 issues.

Jim Savage,
510 Clinton Blvd.
Clinton, MS. 39056-5314

I am sure Robert, and other users of CLAN, appreciate your offer and take you up on it. As I am not personally familiar with Basic compilers, I will have to pass it to the readers. Ok - out there, will someone submit an article on compilers?

- Ed -

UPGRADING TO DOUBLE-SIDED DRIVES

Will you fill us in on how to install double-sided, double density drives into the TRS-80, and the possible modifications, if any on cables, or whatever? There's been a lot of talk about how to install double-sided drives, to replacer the SSDD 180 track drives, on TRS-80 Echonet, but there is a lot of confusion here also! It would be nice if someone was to write an article on this subject, straight and to the point.

Charles T. Neighbors
Longmont, CO.

Let me preface by saying that I am not a hardware hacker. Messing around under the computer's hood is about as inviting to me as sitting down and figuring my taxes. UGH!

However, my Model 4P has had single-sided drives from day one, and I just came across a couple of double-sided drives from a PC, at a good price, so I decided to play mechanic. For a guy who has immense troubles with screwdrivers, it turned out to be relatively easy. All I needed was just that, a 'smaller-sized' phillips-head screwdriver.

My biggest problem was opening the case. I had removed all the obvious screws, but I still could not slide the case off. Not until I figured out that the screws on the 4P handle must also be removed, did I manage to slide the case off and actually have the naked computer sitting in front of me. The drives were held in place by screws. I removed those, unhooked the drive cable to drive :0, as well as unplugging the plug going to the power supply. I then slid out the single-sided drive :0 and replaced with the new double-sided drive, plugging it to the drive cable and the power supply. I repeated this with drive :1. Worked like a charm, so I fastened the drives, replacing the screws, and then slid the case back on.

Nothing to it. I do have two screws left over, but it seems to work fine. Maybe one day I'll figure out where they came from.

I have not yet taken the top off of my desktop Model 4, but I am told that replacing drives is even easier there. Do note that when removing and replacing the cover, be extremely careful that you don't break the back of the CRT.

I will try to get one of the local hardware gurus to write a blow-by-blow account on how to do it - maybe even in English!

-Ed.

MORE TRSDOS 1.3. PATCHING BLUES

I have a problem with the TRSDOS 1.3. patch from vol. 3.1, page 17. I was unable to use it on May 1, 1981 & July 1, 1981 TRSDOS 1.3. system disks. The hangup is the third patch (ADD = 4ED9), which gives the 'STRING NOT FOUND' error. It would appear that there have been some undocumented changes on some disks. I am using a 128K Model 4 with 2 SSDD drives, although I doubt that would have any bearing on the problem.

Using Super Utility, I also found the message 'CAN'T ACTIVATE DUAL' on my disk, just as you did. Also, using SU, I still do not understand how to find an address when displaying files of displaying disk sectors. Somehow, I can't find the explanation in the Super U manual. Please help me.

Leonard Falevitch
Phoenix, AZ

The reason the third patch would not work is very simple. IT IS WRONG. Though I was extremely careful when copying the patch to the TRSTimes master page, I obviously was not careful enough. I made a typo in the third patch. Here is the correction:

PATCH *1 (ADD = 4E9D,FIND = E5C52A,CHG = C3BB4E)

As you can see, the last two digits of the address were inverted. I skimmed through the SU manual and, like you, did not find any reference to the file loading addresses. It is quite possible that SU does not provide this information. How about it, can someone set us straight?

*Now, knowing that overlay *1 loads at 4E00H, I started reading the SU hex listing of record 0. By the time I got to 9D I found the sequence of code the patch is looking for. Thus, adding 9DH to 4E00H, I got the correct address of 4E9DH.*

Sorry about the typo.

-Ed

LESCRIPT WANTED

Enjoy your magazine very much. This is my third year. I have heard of a word processor program called LeScript. If you know of anybody who would like to sell one for a reasonable price, let me know. It should be for Model III.

Oliver J. Jones
144 Poplar Street
Garden City, NY. 11530

LeScript is in my opinion the finest word processor for the TRS-80 and it is reasonably priced. Best of all, it is still supported by the author. Contact the people at:

ANITEK SOFTWARE PRODUCTS
PO BOX 361136
MELBOURNE, FLA. 32936
(407) 259-9397

-Ed

BUGS

How excited I was when I saw that I could speed up my Model 4 in III mode when using SuperScripsit. My DOS is TRSDOS 1.3, and my my version of SuperScripsit is 1.2.08. That sounded all so great that I couldn't wait to get to my computer to type in the speeding up procedure that you recommended on p.18 of the Jan/Feb 1990 issue.

But what a disappointing surprise I had when my Super-Scripsit data was shot to pieces, zap zap zap, making it impossible for me to restore the data that I had so lovingly cherished.

What's wrong, do you think, with me and my 1.3 and 1.2.08 that such a devastating event should occur? It would be a pleasure to hear from you about this? Perhaps other readers might have had the same problem.

James Lewis Lowe
Norwood, PA.

Unfortunately, both the Basic version (listing 3) and the assembly language version (listing 5) of the TRSDOS 1.3 speedup program contained an error. In the Basic version, line 20 is incorrect. The parenthesis after PEEK(&H4210 is listed as a left parenthesis. It should, of course, be a right parenthesis. As written, this program will simply stop at line 20 and report a syntax error. The correct line is:

20 POKE &H4210,PEEK(&H4210) OR 64

The error in listing 5 is somewhat more devious. The seventh and eight lines were listed as follows:

```
LOOP    LD      HL,17408    ;TRSDOS 1.3.'s
        LD      (HL),227    ;internal delay
```

The label LOOP is placed on the wrong line. The correct seventh and eight lines are:

```
        LD      HL,17408    ;TRSDOS 1.3.'s
LOOP    LD      (HL),227    ;internal delay
```

As I don't have a copy of SuperScripsit, I tested the incorrect version of listing 5 by reading and writing a text file with LeScript. No problems, but then LeScript never has. SuperScripsit, I am told by users and ex-users, has a nasty habit of regularly destroying files by mishandling the end-of-file marker.

I am sorry that you lost your valuable date and do hope you had recent backups.

-Ed.

BACK ISSUES

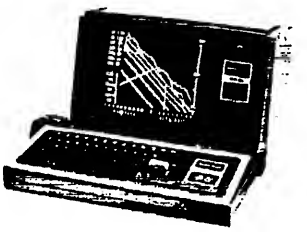
Having heard about you on a BBS, I subscribed about a month ago. I thought I would be receiving a newsletter, you know, a few printed pages with some information about my Model 4. Needless to say, I was impressed when I got my first two issues. Not a newsletter, but a real magazine, and just for the TRS-80. Thanks for keeping my Mod 4 useful.

P.S. Is it possible to get the back issues from 1988 and 1989?

James R. Cobb
Clearwater, FL.

Thank you for the encouraging words. By the time you read this we will have a limited number of back issue sets from 1988 and 1989 (about 100 for each year). The 1988 set has been redone on the laser printer, and is much easier to read than the original dot matrix copies. Each set will be available at $18.00.

-Ed.



Contour

A Graphics Program for Model I & III to Plot Three-Dimensional Figures

By Delmer D. Hinrichs

Have you ever looked at those computer-plotted surfaces and wished that you could make similar plots? If you have a dot-matrix printer with the ability to plot bit-mapped graphics, you can!

The program given here follows the method given by Bob Boothe in 80 Micro, April 1981. However, it is designed to allow for a variety of figures to be plotted, and is written for Epson-type printers. It is easily modified for other bit-mapped dot-matrix printers. The program will plot the figures directly on the printer, or save the figures as a disk data file for later use. Plotting a figure from a disk data file takes only a few minutes, compared to several hours that may be required to calculate the position of each of the 125,000 dots that make up a full-sized figure.

Program

The BASIC Contour program will either print, or save a figure onto disk. For making multiple copies of figures, it is more practical to save a disk data file then print copies from the disk file.

Sample Contours

The simplest Contour is shown in Fig. 1, a hemisphere rising from a flat surface. A slightly more complex Contour is shown in Fig.2, a paraboloid rising from a flat surface. Other shapes available are: bivariate normal (a normal distribution curve rotated about its vertical axis), exponential spire, plus & minus horns, saddle, cross-saddle (a saddle with four low places), monkey-saddle (with three low places), four-way groove/ridge, deep throat, circular waves, and bumpy waves.

These twelve basic shapes can be used to form an unlimited number of different figures, as each individual shape can be varied in where its center is located in both the X and the Y direction, in its magnification in both X and Y directions, and in its vertical (Z direction) magnification; the latter may be positive (upwards) or negative (downwards). In addition, up to seven different shapes may be selected to be added together to form a composite shape.

Choosing Plotting Parameters:

There are several values that must be entered before the program can plot a figure. To avoid confusion, the program suggests suitable values, and defaults to a pre-selected value if you just press <ENTER>. Some trial and error may be needed to get that "perfect" figure. I usually start with a very small trial figure, only 50-100 dot lines.

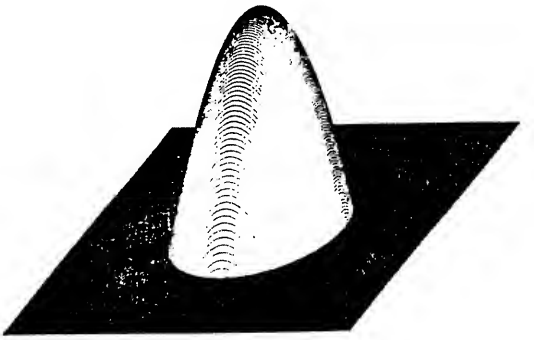
The program first asks whether you want to load a previously calculated figure, or to make a new figure. If you want to make a new figure, it asks whether you want to print it out as it is calculated, or save it to disk. If you save a disk data file, be sure to have enough disk space free. A full-sized figure with 750 lines of dots that nearly fills an 8.5 by 11 page will require 71 grans of disk space. So on a 35-track, SSSD data disk, there is not quite enough room, and you can use only 714 lines of dots for your figure.

Next, the program asks how many lines of dots you wish to print. The Epson printer prints 72 lines per inch, so the 750 lines suggested will print a figure that is about 10.4 inches long. Since the program always plots a square surface, the time required to calculate a figure is proportional to the square of the number of lines of dots you select.

After displaying the size of the field to be plotted (calculated from the number of lines of dots you entered), you are asked to enter the number of shapes to be plotted. Up to

Figure 1

Figure 2



seven shapes may be plotted for each figure. Plotting more shapes gives a more complex surface, but requires more calculation time.

The left margin offset is in the same units as the size of the field. It determines how far across the paper the plot starts. Adjustments may be needed to keep tall shapes from going off-scale.

Next you must set the parameters for each shape:

Set the position of each shape. The X position is crosswise of the sheet, while the Y position is lengthwise of the sheet. While the suggested limits place the center of the shape within the plotted surface, actually the center of the shape may be placed off the plotted surface, so that only its side extensions affect the Contour that is to be plotted.

The X and Y magnifications that you select for each shape determine how broad or narrow the shape is. The Z multiplier for each shape is in the same units as the size of the field. Selection of these parameters will allow you to plot good values; it saves time to try them on a small figure first. Then when the proportions look right, expand the figure to the full size.

A compiled BASIC version of this program will take nearly two hours to calculate a single full-sized figure with 750 lines of dots, but will print such a figure from a disk file in just over six minutes. This is almost full printer speed in the high-resolution graphics mode. BASIC versions (uncompiled) will take eight to ten times as long, making it a good overnight job.

During calculation, the program displays the dot line numbers that it is working on, in groups of eight, to assure you that the program is still working.

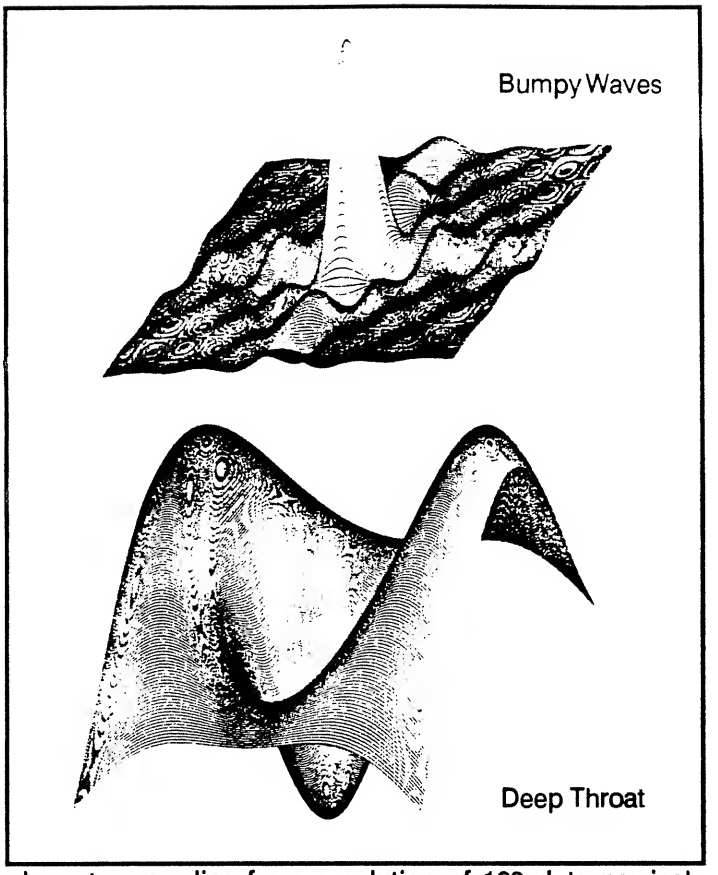
Program Details:

For TRS-80 systems with an Archbold clock control board, the OUT 254,1 statement in program line 580 increases speed. To slow the clock back to normal for disk operation or for the end of the program, the OUT 254,0 statement is used. These statements have no effect if you do not have this board.

The Model III TRS-80 handles characters sent to the printer in a different manner than the Model I, so the program checks whether a Model I or a Model III is being used, and sends the graphics characters properly for either system.

Other Printers

Though it was written for Epson-type printers, this program could be easily modified to work with any dot-matrix printer that can print bit-mapped graphics. Epsons in the bit-mapped graphics mode print eight dots in a vertical line for each character that it receives. Any ASCII character from 0 to 255 prints dots that correspond to which bits are "on." Since there are no excluded Control codes, the printer must be told in advance how many characters it is to interpret as graphics characters. These programs print 960 graphics characters per line for a resolution of 120 dots per inch horizontally. The highest bit, corresponding to ASCII 128, prints the highest dot, while the lowest bit, corresponding to ASCII 1, prints the lowest dot.



Eight horizontal lines of dots are calculated and are saved in the "L(n)" array before printing or saving to disk. For each dot line, the variable NS is set to the ASCII value for that line. If a dot is to be set, NS is ORed with the correct integer from the 960-element "L(n)" array.

If your printer prints fewer than eight dots at a time, the variable K should have fewer that the eight levels that it has in these programs. If the low bit is printed at the top, then K should be set Initially to 0 (instead of 7) and Incremented (instead of decremented) for each new dot line.

For other than 960 dots per horizontal line, reset the value of the variable MM in this program. With a little trial and error, any printer that supports bit-mapped graphics may be accommodated.

Math

These twelve shapes are formed by the use of simple mathematical formulas. These are in program lines 690 to 1010, with each formula identified. The hidden-dot algorithm to give the simulated three-dimensional perspective is explained in Scientific and engineering Problem-Solving with the Computer, by William Ralph Bennett, Jr. (*Prentice-Hall, 1976*), pp. 86 - 90.

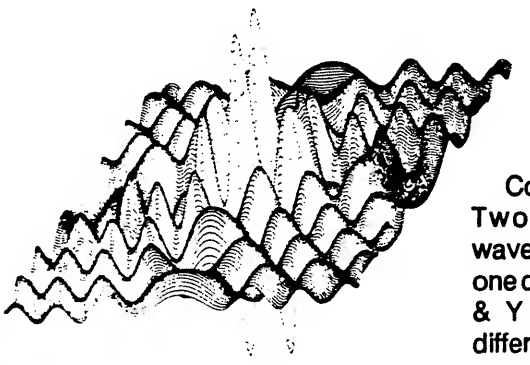
Conclusion

While it is not difficult to plot three-dimensional figures with a bit-mapped dot-matrix printer, it is time consuming. There are too many possible variations to examine all of them. If anyone finds an especially interesting surface, I would like to see a copy. Compiled versions of the program for Model I use are available from the author. If you have a Model III or 4, BASIC or FORTRAN source code is available.

CONTOUR/BAS

```
10 ' CONTOUR/BAS, Version of 15 February 1990
20 ' a 3-D bit-mapped graphics program for Epson 9-pin
printers
30 ' by Delmer D. Hinrichs, 2116 SE 377th, Washougal,
Wn. 98671
40 DEFINT I-N :DIM
B$(255),L(960),XP(7),YP(7),XM(7),YM(7),ZM(7)
50 CLS :PRINTTAB(15)"******  Contour  ******"
60 ID=0 :IL=0 :MM=960 :FS$="CONTOUR/DAT" :PRINT
70 PRINT"Load Contour figure from disk or Make new
(L/M)? ";
80 GOSUB 1280 :IF A$="L" THEN IL=1 ELSE IF A$"M"
GOTO 70
90 IF IL GOSUB 1170 :GOTO 1400 ELSE PRINT
100 PRINT"Send Contour figure to Printer or Disk (P/D)?
";
110 GOSUB 1280 :IF A$="D" THEN ID=1 ELSE IF A$<>"P"
GOTO 100
120 CLS :PRINT"Choose the plotting parameters";
   " (suggested values in parenthesis)" :A$=""
130 INPUT"No. of lines of dots to print (750)"; A$
:LN=VAL(A$)
140 IF A$="" THEN LN=750 :PRINT STRING$(37,25);
CHR$(27); LN
150 PRINT :A$="" :MF=LN*2/3 :IF LN<1 OR LN>1500
GOTO 130
160 PRINT"Then contour is on a";MF;"x";MF;"square field"
:PRINT
170 INPUT"Number of forms to combine on field (1 to 7)";
A$
180 NF=VAL(A$)
190 IF A$="" THEN NF=1 :PRINT STRING$(46,25);
CHR$(27); NF
200 PRINT :A$="" :IF NF<1 OR NF>7 GOTO 170
210 INPUT"Left margin offset (300)"; A$ :LM=VAL(A$)
220 IF A$="" THEN LM=300 :PRINT STRING$(26,25);
CHR$(27); LM
230 IF LM0 OR LM900 GOTO 210
240 FOR I=1 TO NF :CLS
250 PRINT TAB(15)"****** Contour ******" :PRINT
260 PRINT"Form No."; I; " for figure to be plotted:" :PRINT
270 PRINT"Choose which shape this form should have"
:PRINT
280 PRINT"      1. Hemisphere", " 7. Cross-Saddle"
290 PRINT"      2. Paraboloid", " 8. Monkey Saddle"
300 PRINT"      3. Bivariate normal",
          " 9. Four-Way Groove/Ridge"
310 PRINT"      4. Exponential spire", "10. Deep
Throat"
320 PRINT"      5. Plus & minus horns","11. Circular
Waves"
330 PRINT"      6. Saddle", "12. Bumpy Waves"
:PRINT :A$=""
340 PRINT"Which form do you want"; :INPUT A$
:IM=VAL(A$)
350 IF A$="" THEN IM=1
360 IF IM<1 OR IM>12 CLS :GOTO 260
370 CLS :PRINT"Form No."; I; ",  X position (0 to"; MF;
")";
380 A$="" :INPUT A$ :XP(I)=VAL(A$)
390 IF A$=""THENXP(I)=MF/2:PRINTSTR-
ING$(38,25);CHR$(27);XP(I)
400 PRINT"Form No.";I; ",  Y position (0 to"; MF; ")";
410 A$="" :INPUT A$ :YP(I)=VAL(A$)
420 IF A$=""THENYP(I)=MF/2:PRINTSTR-
ING$(38,25);CHR$(27);YP(I)
430 PRINT :IF IM=5 THEN M=MF ELSE M=MF/2
440 PRINT"Form No."; I; ",  X  (width) multiplier ("; M; ")";
450 A$="" :INPUT A$ :A=VAL(A$) :IF A=0 THEN
A=6.28319/M
460 IF A$="" THEN XM(I)=A :PRINT STRING$(45,25);
CHR$(27); M ELSE XM(I)=6.28319/VAL(A$)
470 PRINT"Form No.";I; ",  Y  (length) multiplier ("; M; ")";
480 A$="" :INPUT A$ :A=VAL(A$) :IF A=0 THEN
A=6.28319/M
490 IF A$="" THEN YM(I)=A :PRINT STRING$(46,25);
CHR$(27); M ELSE YM(I)=6.28319/VAL(A$)
500 M=LM+XP(I)-20
510 PRINT :PRINT"Form No."; I;",  Z  (height) multiplier (";
-M; "to"; M; ")";
520 A$="" :INPUT A$ :ZM(I)=VAL(A$)
530 IF A$="" THEN ZM(I)=M
540 NEXT I :CLS
550 DI=1/8192 :K=7 :NC=0 :IF ID=0 GOTO 570 ELSE
GOSUB 1170
560 OPEN"O",1,FS$ :PRINT #1, LN
570 IF ID=0 LPRINT CHR$(27); "@"; CHR$(27); "A";
CHR$(8)
580 OUT 254,1 ' Speed up if Clock Control Board present
590 FOR I=1 TO LN :PRINT"Dot line";I,
600 NS=1 :IF K FOR L=1 TO K :NS=NS+NS :NEXT L
610 JL=MM :JH=0
620 FOR L=1 TO MF
630 IF L>I THEN L=MF :GOTO 1060
640 IX=(I-L)*2 :IF IX>MF GOTO 1060
650 X=IX :Y=L :Z=IX+LM
660 FOR N=1 TO NF
670 X1=(X-XP(N))*XM(N)+DI :Y1=(Y-YP(N))*YM(N)+DI
680 ON IM GOTO 690,720,750,770,790,820,850,880,910,
950,990,1010
690 ' Hemisphere
700 X1=X1/3 :Y1=Y1/3 :ZD=1-X1*X1-Y1*Y1 :IF ZD<0
GOTO 1020
710 Z=Z-SQR(ZD)*ZM(N) :GOTO 1020
720 ' Paraboloid
730 X1=X1/3 :Y1=Y1/3 :ZD=1-X1*X1-Y1*Y1 :IF ZD<0
GOTO 1020
740 Z=Z-ZD*ZM(N) :GOTO 1020
750 ' Bivariate Normal
760 Z=Z-EXP(-(X1*X1+Y1*Y1))*ZM(N) :GOTO 1020
```

```
770 ' Exponential Spire
780 Z=Z-EXP(-(SQR(X1*X1+Y1*Y1)))*ZM(N) :GOTO 1020
790 ' Plus & Minus Horns
800 X1=-X1 :Y1=-Y1
810 Z=Z-X1*Y1*Y1/(X1*X1+Y1*Y1*Y1*Y1)*ZM(N) :GOTO
1020
820 ' Saddle
830 X1=X1/10 :Y1=Y1/10
840 Z=Z-(Y1*Y1-X1*X1)*ZM(N) :GOTO 1020
850 ' Cross-Saddle
860 X1=X1/10 :Y1=Y1/10
870 Z=Z-X1*Y1*(X1*X1-Y1*Y1)/(X1*X1+Y1*Y1)*ZM(N):GOTO
1020
880 ' Monkey Saddle
890 X1=X1/10 :Y1=Y1/10
900 Z=Z+(X1*X1*X1-3*X1*Y1*Y1)*ZM(N) :GOTO 1020
910 ' Four-Way Groove/Ridge
920 XA=ABS(X1)/10 :YA=ABS(Y1)/10
930 Z=Z-((XA-YA)*(XA-
YA)+2*XA*YA/SQR(X1*X1+Y1*Y1))*ZM(N)
940 GOTO 1020
950 ' Deep Throat
960 X1=X1/10 :X2=X1*X1 :Y1=Y1/10 :Y2=Y1*Y1
970 Z=Z-(X2+2*Y2)*EXP(1-X2-Y2)*ZM(N) :GOTO 1020
980 ' Circular Waves
990 W=SQR(X1*X1+Y1*Y1)+DI :Z=Z-SIN(W)/W*ZM(N)
:GOTO 1020
1000 ' Bumpy Waves
1010 Z=Z-SIN(X1)/X1*SIN(Y1)/Y1*ZM(N)
1020 NEXT N :IZ=Z
1030 IF IZ<0 OR IZ>MM GOTO 1060
1040 IF IZ>JH THEN JH=IZ :L(IZ)=L(IZ) OR NS
1050 IF IZ<JL THEN JL=IZ :L(IZ)=L(IZ) OR NS
1060 NEXT L :K=K-1 :IF K0 GOSUB 1330
1070 NEXT I :IF K<7 GOSUB 1330
1080 PR=NC*MM/256 :MM=(1-(PR-INT(PR)))*256
1090 IF MM AND ID GOSUB 1330
1100 IF ID=0 LPRINT CHR$(27); "@"; STRING$(2,13)
1110 OUT 254,0 :CLOSE
1120 PRINT :PRINT"Finally Done!!!"
1130 PRINT :PRINT"Do you want to do another one
(Y/N)? ";
1140 GOSUB 1280 :IF A$="Y" GOTO 50 ELSE IF A$<>"N"
GOTO 1140
1150 CLS :PRINT"That's all, then" :PRINT :PRINT
1160 END
1170 ' Get FileSpec
1180 PRINT :A$=""
1190 PRINT"Data FileSpec now = "; FS$, :INPUT"New = ";
A$
1200 IF A$<>"" THEN FS$=A$
1210 FOR I=1 TO LEN(FS$)
1220 A=ASC(MID$(FS$,I,1))
1230 IF A>95 THEN A=A-32 :MID$(FS$,I,1)=CHR$(A)
1240 NEXT I
1250 IF INSTR(FS$,"/")=0 THEN FS$=FS$+"/DAT"
1260 RETURN
1270 ' Get one character from keyboard
1280 PRINT CHR$(95); CHR$(24); :A$=""
1290 A$=INKEY$ :IF A$="" GOTO 1290
1300 A=ASC(A$) :IF A>95 THEN A=A-32 :A$=CHR$(A)
1310 PRINT A$ :RETURN
1320 ' Save figure on printer or disk?
1330 IF ID=0 GOTO 1580
1340 ' Save bit-mapped graphics characters onto Disk
1350 OUT 254,0
1360 FOR L=1 TO MM
1370 PRINT #1, CHR$(L(L)); :L(L)=0
1380 NEXT L :OUT 254,1 :PRINT :K=7 :NC=NC+1
1390 RETURN
1400 ' Load Disk Data File
1410 ON ERROR GOTO 1430
1420 OPEN"I",1,FS$ :INPUT #1,LN :CLOSE:OPEN"R",1,FS$
:GOTO 1440
1430 CLOSE :PRINT"Data File not found.  Try again"
:GOTO 60
1440 OUT 254,1
1450 FOR L=0 TO 255 :FIELD 1, L*1 AS P$, 1 AS B$(L)
:NEXT L
1460 CLS :PRINT"Loading Data from:  "";FS$;"" :ON
ERROR GOTO 0
1470 LPRINT CHR$(27); "@"; CHR$(27); "A"; CHR$(8);
1480 FOR L=0 TO 960 :L(L)=0 :NEXT L :J=256
1490 NP=LN/8 :IF NP<>LN/8 THEN NP=NP+2 ELSE
NP=NP+1
1500 FOR L0=1 TO NP
1510 FOR L=1 TO MM
1520 IF J=256 OUT 254,0 :GET 1 :OUT 254,1:J=0:IF
L0=1 AND L=1 THEN J=6
1530 L(L)=ASC(B$(J)) :J=J+1
1540 NEXT L :IF L0>1 GOTO 1550
1550 GOSUB 1580
1560  NEXT L0 :GOTO 1100
1570 ' Print bit-mapped graphics characters on 9-pin
Epson
1580 LPRINT CHR$(27); "L"; CHR$(192); CHR$(3);
1590 IF PEEK(84)=1 GOTO 1640 ' Check if Model I or
Model III
1600 FOR L=1 TO MM
1610 IF PEEK(14312)>127 GOTO 1610
1620 OUT 248,L(L) :L(L)=0
1630 NEXT L :GOTO 1680
1640 FOR L=1 TO MM
1650 IF PEEK(14312)>127 GOTO 1650
1660 POKE 14312,L(L) :L(L)=0
1670 NEXT L
1680 K=7 :LPRINT :IF IL=0 THEN PRINT
1690 RETURN
```



Combination: Two circular waves, one up, one down, with X & Y multipliers different

Power programming with the Z-80 alternate registers.

By Jeff Joseph

The Zilog Z-80 processor used in the TRS-80s is gifted with an unusually large number of registers; more than any other 8-bit chip (and enough to rival the Intel 8086/8088). For the most part, however, half of them remain unused by programmers because they exist in an 'alternate' set that require some special (but not difficult) techniques to be accessible.

To appreciate the reasoning for an alternate set of registers we must delve into the origins of the Z-80. It was based on the first successful 8-bit processor, the Intel 8080. The 8080 had the familiar 8/16-bit registers A, HL, DE, and BC, with a set of 78 instructions. The Z-80 capitalized on the 8080's enormous popularity by executing the entire 8080 instruction set, but it's not merely a clone! Zilog added a complete set of secondary registers corresponding to the primary set: A', HL', D E', and BC'. (in addition to other niceties the Intel chips don't provide, even the newest ones, like indexed addressing, block trasfers/searches, and relative jumps!)

The catch is, you can't have access to both sets of registers at the same time. When the Z-80 is powered up, you're in control of the primary set. The Z-80 registers are essentially the same as the old 8080 until you execute one of these two special instructions: EXX and/or EX AF,AF'. The EXX opcode swaps HL, DE, and BC with their alternates. The EX AF,AF' swaps the accumulater and flags with its alternate. Both these are single byte instructions that execute in four clocks, making them very efficient (they execute in three clocks on the 64180, and two clocks on the Z-280). Don't confuse these two exchange instructions with the two other "regular" exchange instructions: EX DE,HL and EX (SP),HL which have nothing to do with swapping in the alternate set.

OK, so you say, what good are these two register sets if I have to keep swapping between them? Well, I'd have to admit it's not as handy as having them all at once, but they can be quite useful. It wasn't possible to simply double the registers because then all the instruction opcodes would need to have an extra bit allocated to address the registers. You can't do that while remaining compatible with the 8080 opcodes. Zilog instead added two flip-flops (single-bit registers) to the Z-80 to switch between the two sets of regs. One flip-flop controls the mapping of the accumulators (manipulated with the EX AF,AF' instruction), and the other flip-flop controls the general purpose register mapping (changed with the EXX instruction). The alternate regs are probably the main reason that the Z-80, to this day, remains the processor of choice in industrial controller applications where you can simply swap in the alternates in event of an interrupt (rather than saving everything on the stack). The Z-80 is the only cost-effective 8-bit chip that can do this.

So how can we use them on the TRS-80? Glad you asked. Suppose your program has some data it needs to 'stash away' someplace where it won't get destroyed, and you need to free some registers for use in the meantime. The classic way of doing this is to PUSH it on the stack. But sometimes your program logic may not allow this, since you need to PUSH more stuff on the stack later and your data will then be buried deep in the stack where you can't get at it. (The Z-280 has a new stack pointer relative addressing mode that comes in handy for this situation) The other classic method is to point at a buffer in memory and stash the data there. You can only do that if register pair HL Is free to do the pointing (unless your data's in A, when you can use BC or DE to do the pointing). The alternate answer (pun intended) goes something like this:

```
EX      AF,AF'
LD      A,D         ; now A'
EX      AF,AF'      ; back to A
```

Here our data is in register D, and we are stashing it in A'. The second exchange opcode switches us back to A, so that data in A' is nice and safe. You can do all sorts of operations on one without disturbing the other. A' is the most useful of t he alternate regs, of course, since you can immediately test the flags and since so many Instructions work with it. Notice also that all six general-purpose regs can Interact with A' In this example, making the data transfer versatile (albeit, only one byte at a time). A' is useful under LS-DOS 6.x because those DOSes trash the accumulator every time you make a call to them. You can instead use A' when invoking an SVC:

```
LD      A,DATA      ; data we want to keep
EX      AF,AF'
SVC     @WHATEVER   ; A' is modified!
EX      AF,AF'      ; switch back to A
```

Here you have to be careful not to rely on A' containing anything in particular later on. You should decide right from the get-go whether you're going to use A' as

storage/counter, or as SVC fodder. The most efficient answer will depend on how much your program uses the SVCs. If you're doing lots and lots of SVC calls throughout the code, it's probably better to not use A' for that since you'd have to sprinkle EXs all over the place, and even at one byte apiece they can add up. However, if your SVC calls are localized in one or two places, it could be worth your while:

```
        LD      A,DATA          ;A contains valid data
        EX      AF,AF'
        SVC     @WHATEVER       ;perform various
        XOR     D               ;operations on A'
        SVC     @AGAIN
        LD      B,DATA
        SVC     @ANOTHER
        LD      E,A             ;put result in a general
        EX      AF,AF'          ;prpose reg & switch back
        ADD     A,E             ;interact with A
```

Now might be a good time to mention that DOS 6.x won't alter the other register set on you, so you needn't worry about SVCs destroying data. However, you should also know that DOS won't save the other register set for you either. This means that if your program will be interrupting another (as in the case of a device driver, filter, interrupt task, PRO-WAM application, or @CMNDR-invokable program), you must preserve the alternates yourself. The program you are interrupting might also be using those other registers, and you must leave them as they were!

As a rule of thumb, these EX instructions should come in pairs. If your code does alot of jumping around you need to be aware of which register set is in context, or you might inadvertantly modify the wrong set. On the other hand, if you are real careful, you might be able to 'share' EXs between routines and save yourself a couple of bytes:

```
RTINE1  EX      AF,AF'          ; not all routines will
                                ; necessarily use A'
                                ; various operations
        JR      EXIT
RTINE2  EX      AF,AF'
                                ; various operations
        JR      EXIT
RTINE3  NOP
                                ; various operations
                                ; not requiring A'
        JP      CONTNU          ; so don't EX AF,AF'
RTINE4  EX      AF,AF'
                                ; various operations
EXIT    EX      AF,AF'
        JP      CONTNU
```

The EXX instruction makes six more bytes of general-purpose storage available to you. The most efficient way to transfer data to them is by using the stack:

```
        PUSH    HL
        PUSH    DE              ; push six bytes on stack
        PUSH    BC
        EXX
        POP     BC
        POP     DE              ; store them in alternate regs
        POP     HL              ; HL', DE', BC' for later use
        EXX                     ; back to primary reg set
```

Programmers working under DOS 6.x will find this handy when dealing with the notorious @VDCTL SVC functions 5 and 6, which destroy all three register pairs:

without using alternates:

```
PUSH    HL
PUSH    DE
PUSH    BC
LD      HL,BUFFER
LD      B,5
SVC     @VDCTL
POP     BC
POP     DE
POP     HL
```

with using alternates:

```
EXX
LD      HL,BUFFER
LD      B,5
SVC     @VDCTL
EXX
```

A savings of four bytes! Additionally, stack usage is minimized which could be important if the rest of the code is using the stack heavily. This example assumes, of course, that you are willing to destroy the alternate register's contents. This same technique works well when using any of the block search/transfer/IO instructions, which also destroy all the registers. How many times have you found yourself constantly loading an address into a register pair to do some pointing? You could load those key addresses into the alternate registers instead, and simply issue an EXX when you want to use those pointers:

without using alternates:

```
LD      HL,POINT
LD      DE,POINT2
LD      A,(DE)
OR      (HL)
;various operations modifying HL,DE
LD      HL,POINT
LD      DE,POINT2
LD      A,(DE)
ADD     (HL)
 ;various operations modifying HL,DE
LD      HL,POINT
LD      DE,POINT2
LD      A,(DE)
CP      (HL)
```

with using alternates:

```
EXX
LD      HL,POINT
LD      DE,POINT2
LD      A,(DE)
OR      (HL)
EXX
EXX
LD      A,(DE)
ADD     (HL)
EXX
EXX
LD      A,(DE)
CP      (HL)
EXX
```

The alternate method might not look much shorter, but it is. Those 16-bit immediate loads are 3 byte instructions, as

compared to the single byte EXX. If you plan ahead you can avoid reloading your pointers over and over. This same strategy applies if you have constants that your code will refer to many times. Memory is usually the place to keep that kind of stuff, but again, you've got to have a register pair free to point to it.

The EXX and EX AF,AF' instructions become even more powerful if we combine them with the other two "regular" exchanges: EX DE,HL and EX (SP),HL. Examine the following macros:

```
EXHLA   MACRO
; exchange HL with HL'
        PUSH    HL
        EXX
        EX      (SP),HL
        EXX
        POP     HL
        ENDM
```

```
EXDEA   MACRO
; ex DE with DE'
        PUSH    DE
        EXX
        EX      (SP),HL
        EX      DE,HL
        EX      (SP),HL
        EXX
        POP     DE
        ENDM
```

The pair of these macros use up 5 and 7 bytes, respectively, which are fairly big as Z-80 instructions go, but they can be very useful to have. No regs are modified, only swapped. These macros return you to whatever set of regs were in context when they were invoked. It may not always be necessary to switch back, however, and depending on your code you might be able to shave off a byte or two. BC and IX/IY are a little harder to get access to, and so these macros use up 9 and 11 bytes, respectively:

```
EXBCA   MACRO
; exchange BC with BC'
        PUSH    BC
                EXX
        EX      (SP),HL
        PUSH    BC
        EX      (SP),HL

        POP     BC
        EX      (SP),HL
        EXX
        POP     BC
        ENDM
```

```
EXIYBCA MACRO
; ex IY with BC'
        PUSH    IY
        EXX
        EX      (SP),HL
        PUSH    BC
        LD      B,H
 ; the Z-280's ex hl,iy
 ;would sure be handy here
        LD      C,L
        POP     IY
        POP     HL
        EXX
        ENDM
```

I've created an entire library of macros like this that perform math and logical functions between the primary and alternate registers as well. Using macros in this manner, you can actually think of all the registers as being available at once, though you will pay a price in bytes with using the alternate set. You can almost always save several bytes if you're willing to destroy some registers, however. Here are some of the possibilities:

```
ADHLDEA MACRO
; add DE' to HL
        EXX
; non-destructive
        PUSH DE
        EXX

        EX   DE,HL
        EX   (SP),HL
        ADD HL,DE
        EX   DE,HL
        ENDM
;note:  sum left in DE for testing
;       original contents of DE left
;       at top of stack
```

```
SBIXHLA MACRO
; IX - HL'
        EXX
; non-destructive
        PUSH    HL
        LD      A,H
; the Z-280 can complement
; hl with one instruction
        CPL
        LD      H,A
        LD      A,L
        CPL
        LD      L,A
        INC     HL
        ADD     IX,HL
        POP     HL
        EXX
        ENDM
; note: A changed
```

```
XRDEDEA MACRO
; XOR DE with DE'
        EXX
; non-destructive
        PUSH DE
        EXX
        EX    (SP),HL
        LD    A,H
        XOR   D
        LD    H,A
        LD    A,L
        XOR   E
        LD    L,A
;note:  result left in HL for testing
;       original contents of HL left
;       at top of stack.  A destroyed.
```

```
LDBCDEAF  MACRO
; LoaD BC,DE' with Flip
EXX
; B<--  E' and C<--  D'
        PUSH    DE
        PUSH    DE
        EXX
        INC     SP
        POP     BC
        INC     SP
        ENDM
```

The Z-280 adds some additional instructions that make the alternate set absolutely awesome! :

```
  JAF   ADDRESS
; jump relative to address if alternate accumulator in use

  JAR   ADDRESS
; jump relative to address if alternate registers in use

  EX    HL,IX       ; swap HL with the index registers (!)
  EX    HL,IY

  EX    A,source
; source = H,L,B,C,D,E,IXH,IXL,IYH,IYL,(HL),(IX+n),(SP+n)

  EX    H,L
; swap high and low bytes of HL
```

In combination with the Z-80's exchange instructions and the many other Z-280 advanced features, the new Zilog chip is truly formidable. Most of us, though, aren't even using the old Z-80 to its fullest potential!

How the TRS-80 Saved my Sanity (and maybe my life...)

By Carol L. Welcomb

It is a rather interesting story, how I became attached to TRS-80 machines. I've always been a writer, but I kept putting off writing this, simply because this story is about ME.

I felt rather dumb about certain computer terms, certain ways of communicating with computers. Well, I guess I've matured, since I no longer care if I say something the rest of you will chuckle about, because you have become my friends.

I became permanently disabled with arthritis in September of 1987. One dumb kidney infection triggered three types of crippling arthritis, not to mention a really terrible attitude problem on my part. I knew I wasn't getting better, but I was unwilling to face the fact that my body would hurt for the rest of my life. I'm thirty five years old, and I didn't want to feel as if I was suddenly seventy.

After six months of being depressed, angry and generally a pain, my best friend's son, Michael, was getting an Amiga 500. (The hostile feelings I have towards Amiga/Commodore could be a whole different article!!) He owned a TRS-80 Model 4P gate array which he was turning over to me.

Well, the only thing I had ever done to the machine prior to actually having it, was to turn it on once and read "The Floppy Disk Drive Is Not Ready" in three languages. Shoot, I thought if I had turned the silly thing on, the least it could do was SOMETHING, ANYTHING... but no! It wasn't ready, and neither was I.

Well, the big Amiga day had finally arrived, and the Model 4P was carted home like a scary treasure. Michael even tossed in the DWP-220, as he was getting a 24-pin printer.

The first thing I did was read the User Manual, feeling like the stupidest person on the face of the earth. After all, I certainly understood nothing in that manual, so I had to be dumb! My children didn't help matters, as I gained the courage to stick the TRSDOS 6.2 system diskette in the drive. They told me to be careful, as one wrong push of a key could blow the whole thing up.

I must mention that, other than using my anytime teller card at the bank, I've had no experience with computers. I have built several electronic components for my stereo, so I did know general circuitry, but this was major league fun. I also need to tell you that I've had trouble with my anytime teller card, too.

From the moment I began to UNDERSTAND the manual, I swore to myself that I would help my fellow TRS-80 users, so no one would ever have to feel as confused as I first did. That's why I've become so involved in corresponding with TRS folks. The computers are gems...the Tandy Users Manual was impossible, at first.

Michael never really enjoyed the 4P. He was more into the fun potential of a computer than its abilities. I soon learned that this little machine could do more than I dreamed it could, and I started cutting back on groceries to save some $$$ for software. I became an avid reader of all the magazines I could get my hands on, mostly by purchasing older, used magazines which dealt with our machines.

Reading was not enough, since I was learning from others how useful and fun these TRS-80 computers were. I had to learn programming, BASIC, and anything else which looked interesting.

Then one day, IT happened. My motherboard had a stroke. That's when I also learned how to replace chips, and look for bargains in the TRS-80 market of hardware.

I've made lots of friends throughout the USA and Canada, due to a faulty board. The absolute best thing is that I've done some serious soldering, add-ons, hi-res (a Micro-Labs and a Radio Shack...in separate computers).

I've learned through my own experience, and from the great letters I read in TRSTimes and CN80, that Radio Shack is not the place for me to take my TRS-80 if it is sick.

I've learned to be patient, and someone comes through for me. I felt like a part of a little world of nice, concerned people who will help you, write to you, tell you how to fix something. Best of all, there are so many people who will say that they know someone who had the same problem...and here's their address.

The ONE and ONLY time I ever mentioned a TRS-80 to a sales clerk at Radio Shack's Computer Center, I received the strangest look. Of course, I love my machines, and I'm partial to them, but this clerk really made me mad. So I started quoting part numbers, and used intimidation, until

I got the answers I wanted. Then I left without ordering anything, knowing I could find what I was looking for through my circle of friends.

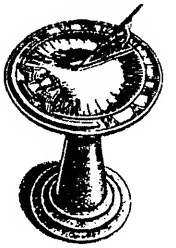
I am in the process of adding a MEG of RAM to my 4P. I do everything a little bit at a time, since I cannot afford to just plunk down big bucks at once. I've built a swell library of TRS-80 literature, and I now feel fairly confident when I tackle a new project. I've gotten quite technical about these machines, which actually was the easiest part. The RUNNING of the machines was my early downfall, but it's pretty much conquered (as of today).

I would like to thank my many friends who helped me acquire the computers/software/books/magazines I am fortunate enough to own. If I have a painful day, I actually don't pay much attention to it, when I'm doing something with the computers. It was my body that got screwed up, but thanks to the TRS-80 computers, my mind is able to continue. It may sound odd to some people, but I feel I owe a lot to these machines, and to the kind people who still love them as much as I do.

Carol Welcomb lives in the woods with her best friend, three children, 4 dogs, 8 cats, 2 tarantulas and 3 snails. She enjoys tearing broken things apart, and feels dandy when she can make things work again (not to mention the thrill of not having any leftover parts). Carol can be reached at: 11161 Edgerton N.E. Rockford, MI 49341

ATRSCLCK/BAS for Model 4

```
10 ' ATRSCLCK/BAS, BY CAROL L. WELCOMB
20 ' BASED UPON A NEUROTIC COMPULSION
25 WIDTH 255
30 DEFSTR A-G,I-L,N-R
40 A=CHR$(48):B=CHR$(49):C=CHR$(50):
D=CHR$(51):E=CHR$(52):F=CHR$(53):
G=CHR$(54):I=CHR$(55):J=CHR$(56):
K=CHR$(57):L=CHR$(128):Q=CHR$(34)
50 N=CHR$(26)+STRING$(6,24):
R=CHR$(26)+STRING$(4,24)
60 AA=STRING$(6,A):
AB=STRING$(2,A)+STRING$(2,L)+STRING$(2,A)
70 BB=STRING$(6,B):
BC=STRING$(2,L)+STRING$(2,B)+STRING$(2,L):
BD=STRING$(4,B)+STRING$(2,L)
80 CC=STRING$(6,C):
CD=STRING$(3,L)+STRING$(3,C):
CE=STRING$(3,C)+STRING$(3,L)
90 DD=STRING$(6,D):
DE=D+STRING$(3,L)+STRING$(2,D):
DF=STRING$(2,L)+STRING$(4,D)
100 EE=STRING$(6,E):
EF=STRING$(2,E)+STRING$(2,L)+STRING$(2,E):
EG=STRING$(4,L)+STRING$(2,E)
110 FF=STRING$(6,F):
FG=STRING$(3,F)+STRING$(3,L):
FI=STRING$(3,L)+STRING$(3,F)
120 GG=STRING$(6,G):
GI=STRING$(2,G)+STRING$(4,L):
GJ=STRING$(2,G)+STRING$(2,L)+STRING$(2,G)
130 II=STRING$(6,I):
IJ=STRING$(4,L)+STRING$(2,I)
140 JJ=STRING$(6,J):
JK=STRING$(2,J)+STRING$(2,L)+STRING$(2,J)
150 KK=STRING$(6,K):
KL=STRING$(2,K)+STRING$(2,L)+STRING$(2,K):
KR=STRING$(4,L)+STRING$(2,K)
155 QQ=STRING$(4,Q)+R+STRING$(4,Q)
160 P(0)=AA+N+AA+N+AB+N+AB+N+AA
+N+AA
170 P(1)=BC+N+BD+N+BC+N+BC+N+BC
+N+BB
180 P(2)=CC+N+CC+N+CD+N+CE+N+CC
+N+CC
190 P(3)=DD+N+DE+N+DF+N+DF+N+DE
+N+DD
200 P(4)=EF+N+EF+N+EE+N+EE+N+EG
+N+EG
210 P(5)=FF+N+FF+N+FG+N+FI+N+FF
+N+FF
220 P(6)=GG+N+GG+N+GI+N+GG+N+GJ
+N+GG
230 P(7)=II+N+II+N+IJ+N+IJ+N+IJ+N+IJ
240
P(8)=JJ+N+JK+N+JJ+N+JJ+N+JK+N+JJ
250 P(9)=KK+N+KL+N+KK+N+KR+N+KL
+N+KK
255 PRINT CHR$(15):CLS
260 P=RIGHT$(TIME$,8):H1=VAL(MID$(P,1,1)):
H2=VAL(MID$(P,2,1)):M1=VAL(MID$(P,4,1)):
M2=VAL(MID$(P,5,1)):S1=VAL(MID$(P,7,1)):
S2=VAL(MID$(P,8,1))
270 PRINT@(9,8),P(H1);:
PRINT@(9,16),P(H2);:PRINT@(9,24),QQ;:
PRINT@(13,24),QQ;
280 PRINT@(9,30),P(M1);:
PRINT@(9,38),P(M2);:PRINT@(9,46),QQ;:
PRINT@(13,46),QQ;
290 PRINT@(9,52),P(S1);:PRINT@(9,60),P(S2);
300 GOTO 260
```



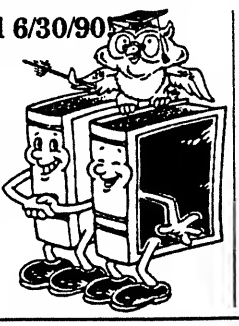
Our new LS-DOS 6.3.1 release has a little something for everyone

- ☆ The DATE command, "Date?" prompt on boot, and the @DATE SVC now support a date range of 32 years; from **January 1, 1980 through December 31, 2011.**
- ☆ **Enable or disable the printer time-out** and error generation with SYSTEM (PRTIME=ON | OFF)
- ☆ Customize the display of the time field in the DIR command to display **12-hr or 24-hr clock time** with SYSTEM (AMPM=ON | OFF).
- ☆ Both ASCII and hexadecimal display output from the LIST command is **paged a screen at a time**. Or run it non-stop under your control.
- ☆ MEMORY displays (or prints) the status of switchable memory banks known to the DOS, as well as a **map of modules** resident in I/O driver system memory and high memory.
- ☆ Specify SYSTEM (DRIVE=d1,SWAP=d2) to **switch drive d1 for d2**. Either may be the system drive, and a Job Control Language file may be active on either of the swapped drives.
- ☆ The TED text editor now has commands to **print the entire text buffer**, or the contents of the first block encountered. Obtain directories from TED, too!
- ☆ Have extended memory **known to the DOS**? The SPOOL command now permits the BANK parameter entry to range from 0-30 instead of 0-7.
- ☆ **Alter the logical record length** of a file with "RESET filespec (LRL=n)"
- ☆ Specify "RESET filespec (DATE=OFF)" to restore a file's directory entry to the old-style dating of pre-6.3 release. Specify "RESET filespec (DATE=ON)" to establish a file's directory date as that of the **current system date and time.**
- ☆ Felt uncomfortable with the *alleged* protection scheme of 6.3? **LS-DOS 6.3.1 has no anti-piracy protection!** MISOSYS trusts its customers to honor our copyrights.
- ☆ Best of all, **an LS-DOS 6.3.1 diskette is available as a replacement disk for $15** (plus $2 S&H in US). There's no need to return your current master.
- ☆ The 6.3.1 diskette comes with a 30-day warranty; written customer support is available for 30 days from the purchase date. Versions for the Model 4 and Model II/12 are available. **If you do not already have an LS-DOS 6.3.0, order the 6.3.1 Upgrade Kit with 90 days of customer support for $39.95** (+$2 S&H).

MISOSYS, Inc.
P. O. Box 239
Sterling, VA 22170-0239
703-450-4181 [orders to 800-MISOSYS (647-6797)]

MISOSYS SPECIALS OF THE MONTH

DoubleDuty — 30% off until 6/30/90



DoubleDuty divides your 128K TRS-80 Model 4 computer's memory into three complete and independent partitions. Two partitions each operate as if they were ther own 64K Model 4. The third can be used to execute DOS library commands. If you thought you needed another computer, think again. With DoubleDuty, you can now have two for the price of one! **Just $34.97 (+$2S&H)!**

GO:CMD

Here's a collection of programs designed to provide additional utility for your computer operation, and are rewritten for Model 4 LS-DOS 6.3. You get FASTBACK and FASTREAD for hard disk large file archive/restore; PRO-CESS to manipulate executable command files; COMP to compare two files or disks; FED2 to investigate and zap disk or file sectors on a full-screen basis; IFC updated with new features for interactively copying, moving, renaming, deleting, and invoking files; ZCAT for cataloging 6.3 disks. Revised documentation is printed in a convenient 5.5" by 8.5" format.**50% off until 06/30/90; Just $29.98 (+$4S&H)!**

HINTS & TIPS

FROM BASIC TO DOS
(THE EASY WAY)
Model 4 LS-DOS 6.3.0 & 6.3.1
By Lance Wolstrup

Leaving Model 4 Basic and returning to DOS takes seven keystrokes. You type **SYSTEM** and press <ENTER>. However, beginning with LS-DOS 6.3.0, and continuing in 6.3.1, there is an easier way. While you may still use the SYSTEM command, you can now simply type **!** and press <ENTER> and, whammo, there you are, back in DOS. Only two keystrokes.

Just as the familiar **?** is shorthand for PRINT, **!** shares the same token as SYSTEM, and can therefore be used from immediate mode, or even from inside a Basic program.

Try this very short program:

10 ! (DIR :0)

Run it, and you will get a directory of drive :0, and return to Basic. When you list the program it has been changed to this:

10 SYSTEM (DIR :0)

I love it!

LDOS 5.1.4.
By Dennis Burkholz

Older versions of the LDOS library command DIR always behaved more like a CAT command; that is, it displayed the filenames in a multiple column format, omitting all other information. A 'normal' directory was available by adding the (A) parameter: DIR :1 (A).

LDOS 5.1.4. defaults to the 'normal' directory display of one filename per line, followed by the essential information of the file. It is still possible to get the short directory. Just add the parameter **(A=N)**. For example: **DIR :1 (A=N)**

MULTIDOS
By David Welsh

One thing I really hate about TRSDOS 1.3. is the nonstop scrolling DIR. I know you can stop it if you are quick, but I'm not. In Multidos you can scroll the DIR a line at a time by pressing the space bar. Try it.

TRSDOS 1.3.
By Per Svengaard-Nielsen

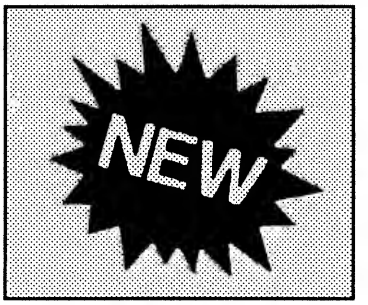
All the other Model III operating systems have utilities to transfer files **FROM** TRSDOS 1.3. to their particular system. How does one transfer a file back **TO** TRSDOS 1.3.?

From the non 1.3. DOS, format a single sided, single density disk and copy all the files you wish to transfer to it.

Boot up TRSDOS 1.3. and use the CONVERT command to copy the files to a 1.3. disk.

Editor's note: If you have TRSDOS 1.5 (available from TRSTimes) you need not go through the above steps, as you can copy directly using the TRSDOS 1.5. library command CPY.

Is there a NEWDOS after TRSDOS 6.3?
By Roy T. Beck



No, there is no NEWDOS after TRSDOS 6.3, but there sure as heck is a NEW TRSDOS, and it is LS-DOS Release 6.3.1.

I know that's true because this is being written on my own personal, legal copy of it, direct from Roy Soltoff, AKA MISOSYS.

It has some nice, new features, and in addition clears up the long-simmering question about copy protection on the 6.3 version.

First, the major items of interest about it:

- There is no copy protection
- Dating has been extended to cover 1980-2011
- Soltoff/MISOSYS is now the owner of TRSDOS/LSDOS 6.3.1
- Easy reconversion of a file's date structure
- SWAP parameter has been added to the SYSTEM command
- TED has been enhanced
- The SPOOL command has been modified to permit the BANK parameter entry to range from 0-30 instead of 0-7

The actual changes are covered in a two page instruction sheet, which is notable for its brevity. Some of the change descriptions will require a little research for complete understanding, at least by me.

Installation of the new release is the same as installation of 6.3; use the same procedures. You will have to SYSGEN again to keep all your special setups, including your existing hard disk configuration. No big deal, just remember to do it. This is all covered in the installation procedure for 6.3.0, so dig it out and review it.

All copyright and welcome messages now show MISOSYS, Inc. where Logical Systems used to be. This is the tipoff that Roy now owns the DOS, lock, stock and barrel, and is no longer a "licensee" from Logical Systems. He can now say, and does, that there is no "copy protection" scheme in 6.3.1, although he still has to be a little fuzzy about 6.3.0. Incidentally, this is stated in his advert, but not in the instruction sheet. I trust him, though, and believe there is no protection scheme; further all of his previous comments have strongly hinted there never was any, he just wasn't legally allowed to say so. Therefore, buy 6.3.1, and use it with confidence. Speaking of buying, his price for the new release is only $17, including shipping. Quite a reasonable price, I believe. His phone number is 800-MISOSYS, so just pick up the phone and dial. He accepts plastic.

Since the date structure now runs to 2011, I believe some of us will expire before the new date system does! I personally will be age 81 by then, but I expect to be still pounding this 4P at that time.

Some other, minor goodies include a 12 or 24 hour clock time in the directory display. This only applies to the directory display, not the clock available on the screen display.

DISKCOPY will no longer ask you to put a system disk in drive :0 before terminating. It will check, and if you have a SYSTEM disk there, it will use it without asking.

The DO command will now search for a drive without a write protect tab to write its output SYSTEM/JCL file. This will avoid aborting the function if you have a write protect on your drive :0, which is the case on all earlier versions.

The command ID will now bring up the answer "No service contract". There is no phone support. I believe this means if you buy a separate service contract, you will receive a disk with a customer ID No. on it.

The LIST command now defaults to paged display; it won't scroll off the screen before you can stop it! However, if you prefer the scrolling, you can add the parameter (NS).

A file can be dated either in the old or new mode by use of the RESET command with DATE = ON for the new system, and DATE = OFF for the old. This will allow easy transfer between DOS's, especially TRSDOS 1.3. When you set a file to the "old" system, the "user" password field will be set to blanks. No more denial of access due to a new-style date in the old field. Thanks, Roy.

SYSGEN can now be done within a JCL file. It used to balk at this. The JCL will be temporarily suspended during the SYSGEN.

You can now SWAP drives, including the system drive, with the command SYSTEM (DRIVE = d1, SWAP = d2). Either drive may be the system drive, and a JCL file may be active on either one of them. I believe DOSPLUS has long had the SWAP portion of this capability, although I don't know about the active JCL file, so Roy has caught up there. (Ron Malo has mentioned this feature as being one of the reasons PROSOFT liked to use DOSPLUS).

TED has had a lot of decorations added, including the ability to print the screen with CTRL-P.

The owner password for all the system files has been changed from LSIDOS to SYSTEM6. I am sure this is just to emphasize the fact that MISOSYS now owns the DOS.

The end-of-file pointer for MODELA/III has long had an erroneous value in the directory. This has now been corrected.

There are many more changes, too many to cover in this essay, and some I don't even understand. Get your own copy and start digging.

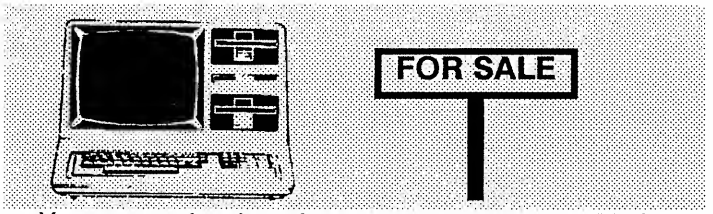
In summary, Roy is still with and supporting us; we must do the same for him. The first order of business is therefore for all of us to buy our own personal copy of 6.3.1, and don't hand any freebies around! For $17, none of us should be so cheap, and Roy needs our support.

What about LDOS? Is there going to be a 5.3.1 release for it? Roy told me there will be, but another project has priority. What project is that? It is the tape backup drive for our hard disks. For a preview, read the ad for the QUIC-40 in recent issues of The MISOSYS Quarterly, which all of you should subscribe to, as it is a gold mine of TRS information. The QUIC-40 is the MSDOS version, but the hardware will probably be identical for the MODs 3 and 4; only the software will be different, I believe. Prices will be about the same for the TRS version.

In any event, support MISOSYS! Roy deserves it and we need Roy. It is that simple.

HOW TO SELL YOUR TRASH

By Eric Bagai



You say you just bought a 25 megahertz laptop Bimbette from Whiz-Bang Peripherals, and now you need some cash to make the next payment? And so you took your old Trash 80 to a user group to sell it and they looked at you funny because they all had Macs? And then you took it to a swap meet and they offered you $25 for the "dumb terminal," and that was the best offer you got? Is that's what's troubling you, Bunky?

You have been trying in the worst possible places. The user group is no good, because anybody who might be interested in a Model 4 has at least two of them already. A swap meet is even worse, because everybody is either a vendor or wants a gold-plated dream for two cents on the dollar. Think about it: why do you go to a swap meet?

You should be able to get $300 for a Model 4, $350 for a Model 4D or 4P, $200 for a Model III, and $150 for a Model I (but only if you nail down all the cable connections and mount all the parts in a case). A Model II/12 can get $300 if you have the right software. A working printer should add $100, and a modem or a hi-res board gets another $50. (The exception is a hard disk, for which a TRS hacker will pay big bucks.) Why $300 and not $500? This is called price positioning: It has to be priced below the PC clones, but it's certainly better than a Timex or a Tommy Tutor. After all, your machine can do more than an IBM 360 could, and the 360 was the mainstay of American industry for almost two decades.

However, you must cut these prices in half if you don't approach this in the right way. You can't just sell a computer--as they say in the advertising biz, you have to sell a solution. To sell a solution, try to imagine the things that people wish they could do. They want to have their own business. They want to manage their finances successfully. They want to win the lottery (and not by blind luck like all the suckers; they'll win it by their own skill and natural cunning). They want to meet people who will like them and think they are intelligent. They want to confirm that they are someone special. They want all the things you and I want; and if they think that having a computer will get it for them, then they just might buy yours. (Next issue we can discuss the ethics involved--this month let's just sell your machine.)

The best place to sell your computer is where the audience is very large and diverse and not terribly sophisticated. Therefore, the worst place (aside from a user group or swap meet) is on a BBS, in the Computer Shopper, TRSTimes, or in the Interface. The best place is in your local newspaper's want-ads. Many areas also have a weekly paper that is all want-ads. In southern California there is The Recycler, and a slew of local publications like The Penny Shopper and The Anaheim Horsetrader. The ad rates in newspapers are not too expensive; the Recycler doesn't charge for ads at all.

The standard ad says something like:

> TRS-80 4P, 128K, DS/drvs, 8 MHz
> 20 MB HD, amber hi-res, intnl
> modem, softwr/docs. $400 obo.
> Ask for Tommy (714) 988-3838.

Even though this is a lovely package at a great price, the ad is worthless. It is incomprehensible to anyone except another TRS owner. To your intended customer it sounds like a kid trying to dump his Pong setup.

What your customer wants to see is:

> COMPUTER BUSINESS SYSTEM: accounting,
> spreadsheet, word processing. Complete
> documentation and tutorials. $300 firm.
> Mr. Johnson (714) 988-3838 after 6pm.

Of course you must actually have what you advertise: a functional payables/receivables package and general ledger, maybe Visicalc, and a decent word processor. Deskmate is a real winner here.

Another good one is:

> PROFESSIONAL WORD PROCESSING SYSTEM.
> For business, writer, or student; with
> all training manuals & spelling/grammar
> checker. Fast printer, large screen &
> good keyboard. $350 firm. (213) 991-1919

This package could be a single-drive Model III, an Epson MX80, Electric Webster, and the original Scripsit with its training tapes and stick-on labels. Throw in a copy of Hypercross and you can even say "IBM text-compatible." Any

advance on this setup may not get you more cash, but it will make your offer that much more solid. Keep in mind that you are not selling junk, you are not lying, and you are not promising something beyond the capability of the computer. Of course it isn't as fast, or powerful, or flashy as a '386 with super-VGA and an 80 Meg drive. But it isn't as expensive either -- and that's what you're selling: ability to do the job but without all the bells and whistles.

Try thinking of how each of the following lead-ins is absolutely true, given the right software and peripherals.

SUPER GAME COMPUTER

TELECOMMUNICATIONS SYSTEM

BEAT THE LOTTERY WITH YOUR OWN COMPUTER

TAP 2,000+ DATA BASES & LIBRARIES FROM YOUR HOME

NEW-AGE COMPUTER: BIORHYTHMS, I CHING, TAROT

BUY YOUR SECRETARY A COMPUTER SHE CAN UNDERSTAND

COMPUTER DATING WITH A REAL COMPUTER

AT LAST! A COMPUTER FOR THE REST OF THE REST OF US

RENT-TO-OWN YOUR OWN COMPUTER

or:

COMPUTER BLOWOUT!
NOT AVAILABLE IN ANY STORE!
WE BOUGHT THE ENTIRE STOCK!
LIMIT ONE PER FAMILY!
(Tandy employees and relatives not eligible)

and my favorite:

MUSEUM OVERSTOCK: ANTIQUE COMPUTERS USED BY ORIGINAL HACKERS

Everybody is afraid of getting ripped off, and there are simple things you can do to minimize this danger. You can also screen your customers to make sure that they qualify as sane, reasonably intelligent, and able to pay. You do this screening on the phone, not at the front door. Have a list of questions ready to ask your customer--and ask them of everyone. Ask what they want to use the computer for, and follow up with more questions about their reasons for wanting a computer. If they are genuine, they will talk themselves into buying your machine--if they aren't, they won't have much of anything to say. Do they need IBM compatibility, and if so, why and how much? Have they ever used a computer before and what kind was it and what software did it use? If they have never used a computer, have they ever programmed a VCR to record a week's worth of shows on different channels? Can they set the alarm on their digital watch? (There is no point in dealing with a computerphobe, or an idiot.) Do they know anyone who has used this type of computer and who can help them? Can they bring cash or a money order for the exact amount--no checks, or at least no counter checks or third party checks? Have them show up at a specific time of your choosing. If you don't think they will be able to benefit from your machine, tell them so. Please remember that politeness does not require you to let people rob you--and you are not obligated to give out your address to anyone. Of course you will be wary of people you don't like, but also watch for people that are exceptionally nice. A "good" con man is probably the nicest person you'll ever meet. So whom can you trust? Somebody like you: just ordinary folks; real people.

When he shows up, be sure to tell your customer where you get the computer serviced. This informs him that you have taken good care of it, and that he should call the service place, and not you, when things go thump. Also establish your return policy: no returns. It works now and there's no reason it shouldn't continue to do so unless it is dropped or cheese is put in the drive doors. Demonstrate the basics of the programs that he said (on the phone) he was interested in. Show him previously run, working printouts of those programs, and demonstrate the tutorials. Give him written information about local user groups, bulletin boards, software sources, and user magazines. And tell him that he can get personal training and general hand-holding there (but only if it is true!) If you agree to train him, be sure he understands that only the first four hours are free, and make sure that all such training occurs at his place.

The software should be neatly labeled and the documentation should be in a binder. If you've mislaid the originals, you can have your xeroxes covered and comb-bound by your local quick-printer. If you still have the original disks and docs, so much the better. If your DOS doesn't respond to the current date, then disable the date query. Everything you advertise should work off the same DOS. You should refer to the rest of the (thoroughly dusted and neatly boxed) stuff as, "programs you will find useful after you are comfortable with the basic software." Do not admit to the existance of Newdos 80. Look professional: five minutes with some Simple Green or Formula 409 does wonders. (On the machine case, silly, not on you.) It may just be an old fuzzy friend to you, but it is a miracle of modern precision and technology to your customer.

Oh, yeah: don't let them see your new computer.

ASSEMBLY 101

Z-80 without tears

By Lance Wolstrup

This installment is the continuation of the TRSLABL4 program for Model 4 which we started in the last issue. As a formality, to all the new readers, let me just say that the program listing in this article will NOT work by itself. It must be typed in along with the previous listing.

OK, let's all get out our editor/assemblers and get to work.

TRSLABEL 4 - part 2

The first half of TRSLABL4/SRC, listed in the last issue, constituted the main body of the program and was, hopefully, explained properly then. The reason that this portion of the program will not work by itself is, of course, that it contained numerous CALLs to non-existing subroutines. This is what this half of the program is all about: SUBROUTINES.

From programming in Basic, you are undoubtably familiar with the GOSUB command and its usefulness. My rule of thumb is that if I have to write any code more than once, I will usually write the code as a subroutine instead, and then GOSUB (if in Basic), or CALL (if writing assembly code).

A minor exception to this rule occurs when I write assembly for the Model 4. Here I will usually include all SuperVisor Calls as subroutines, even if they are only used once. Why? Simply because it makes it easier for me to write the code. Also, I always try my best to have the subroutines listed in alphabetical order. It makes them easier to find, both on the screen and on a print-out. Sometimes it doesn't work out that way - but I try.

The first subroutine (lines 02140-02160) is the clear screen routine. We use the SVC, by loading register A with 105 and then executing the routine with RST 40. Keep in mind that this SVC was not implemented until TRSDOS 6.2. If you are working with an earlier version of the DOS, the screen will not clear.

Next is the DSP routine (lines 02180-02220). It is the SVC to send one character to the screen. The tech manual explains that this SVC alters register DE, thus we make sure to PUSH DE on the stack before the routine is executed. Then, after completion, we POP the original value back to DE and RETurn to the caller.

The DSPLY SVC (sends a string of characters to the screen) works much the same as DSP. Here, though, we make sure to preserve the value of register HL.

The above routines are convienently provided for us by the authors of the DOS. Unfortunately, not everything we

```
02110 ;
02120 ;subroutines
02130 ;
02140 CLS       LD      A,105           ;@cls
02150           RST     40
02160           RET
02170 ;
02180 DSP       PUSH    DE
02190           LD      A,2             ;@dsp
02200           RST     40
02210           POP     DE
02220           RET
02230 ;
02240 DSPLY     PUSH    HL
02250           LD      A,10            ;@dsply
02260           RST     40
02270           POP     HL
02280           RET
02290 ;
02300 ERASE     LD      C,15            ;cursor off
02310           CALL    DSP
02320           LD      BC,0520H        ;b=counter, c=chr
02330 ELOOP1    PUSH    BC              ;save counter & chr
02340           CALL    LOCATE          ;position cursor by HL
02350           LD      B,E             ;get loop counter
02360 ELOOP2    CALL    DSP             ;erase horizontally
02370           DJNZ    ELOOP2          ;continue until done
02380           POP     BC              ;restore loop counter
02390           INC     H               ;bump line
02400           DJNZ    ELOOP1          ;continue until done
02410           LD      BC,0531H        ;b=loop counter, c=chr
02420           LD      HL,0412H        ;cursor position (4,18)
02430 ELOOP3    PUSH    BC              ;save counter & chr
02440           PUSH    HL              ;save cursor position
02450           CALL    LOCATE          ;position cursor
02460           CALL    DSP             ;display chr
02470           INC     L               ;move the
02480           INC     L               ;cursor twice
02490           CALL    LOCATE          ;to the right
02500           LD      C,'-'           ;c=chr
02510           CALL    DSP             ;display "-"
02520           POP     HL              ;restore old cursor pos
02530           INC     H               ;bump to next line
02540           POP     BC              ;restore counter & chr
02550           INC     C               ;bump chr
02560           DJNZ    ELOOP3          ;continue until done
02570           RET                     ;return to caller
02580 ;
02590 INKEY     PUSH    BC              ;save max key strokes
02600 ILOOP1    CALL    DSP             ;display underline chrs
02610           DJNZ    ILOOP1          ;continue until done
02620           LD      E,0             ;e=chr count
02630 INKEY1    CALL    LOCATE          ;position cursor
```

want to do in a program is as simple as that. Some routines we have to write ourselves. Such is the case of the next one:

ERASE is called from line 00980 (last issue). We want to erase the field headings, so before the routine is entered we have positioned the cursor @(4,0), the first line of field headings. We also loaded register E with the value 22, which is the length of the longest field heading. Then we CALLed the ERASE routine. By loading register C with 15 (line 02300) and then CALLing DSP, we are doing the same as PRINT CHR$(15) in Basic, turning the cursor off.

Line 02320 loads register BC with 0520H. Rather than thinking of this as BC=0520H, think of it as register B contains 05H (5) and register C contains 20H (32). B is the counter for how many times we have to loop to erase each of the 5 lines of field headings. C holds the character 20H, which is the space character. We will use it to erase the lines.

Line 02330 begins the outer loop (ELOOP1). First, we save the value of BC on the stack as register B is needed later. Then, we make sure the cursor (even though invisible) is where we want it. We do this by CALLing the LOCATE subroutine, which positions the cursor according to the value of HL. Register HL was defined just before we entered the ERASE routine and, since we haven't modified HL, the cursor remains @(4,0).

Line 02350 loads B with the value of register E (remember, it is 22) for use as the counter for the inner loop.

Line 02360-02370 is the inner loop. The routine simply displays 22 spaces next to each other, thus erasing the field heading. By CALLing DSP 22 times, we accomplish this.

We now need to restore the counter for the outer loop (the one determining how many lines to erase), so we POP BC (line 02380). We also need to move the cursor to the next screen line. Line 02390 INCrements the value of register H, which will move the cursor to where we want it when we get back to the CALL LOCATE in line 02340.

Line 02400 (DJNZ ELOOP1) decrements B and, if B is not 0, goes back to ELOOP1, starting the whole process over.

By the time the program falls through to line 02410, we have erased the field headings on each of the five lines. Now we can display the field ID numbers on the screen, and that is exactly what ELOOP3 will do.

First, however, we load BC with 0531H. Again, think of this as B=05H (5), and C=31H (49). Register B is the counter to tell us how many times to loop, and register C holds the character to display (CHR$(49) is the number "1").

Next, in line 02440, the cursor position is stored in register HL (H=line 4, and L=column 18).

Then we go into the ELOOP3 routine (line 02430) where BC is PUSHed onto the stack, thus saving the loop counter and the character to display. Following that, the location of the cursor (in register HL) is also PUSHed to the stack.

In line 02450 the cursor is actually positioned, as we CALL the LOCATE subroutine. Then, by CALLing the DSP routine (line 02460), the character in register C is displayed on the screen. At this point we have displayed the number "1".

```
02640           LD      C,14        ;turn cursor on
02650           CALL    DSP
02660           CALL    KEY         ;get keystroke
02670           CP      13          ;is it <ENTER> ?
02680           JR      NZ,INKEY2   ;no - go on
02690           LD      (IX),A      ;yes - store CR
02700           LD      C,15        ;turn cursor off
02710           CALL    DSP
02720           POP     BC          ;restore max keystrokes
02730           LD      A,B         ;copy to a
02740           SUB     E           ;sub # of chrs used
02750           INC     A           ;make sure at least 1
02760           LD      B,A         ;copy to loop counter
02770           LD      C,32        ;c=chr$(32)
02780 ILOOP2    CALL    DSP         ;loop to erase unused
02790           DJNZ    ILOOP2      ;portion of the line
02800           RET                 ;return to caller
02810 INKEY2    CP      8           ;is it LEFT ARROW  ?
02820           JR      NZ,INKEY3   ;no - go on
02830           LD      A,E         ;yes - chr counter to a
02840           OR      A           ;is counter=0  ?
02850           JR      Z,INKEY1    ;can't backspace if 0
02860           DEC     L           ;move cursor left
02870           CALL    LOCATE
02880           DEC     E           ;chr count= chr count-1
02890           LD      C,95        ;replace current chr
02900           CALL    DSP         ;with chr$(95)
02910           DEC     IX          ;dec buffer pointer
02920           JR      INKEY1      ;get next keystroke
02930 INKEY3    CP      14          ;is it ctrl-n ?
02940           JR      NZ,INKEY4   ;no - go on
02950           POP     BC          ;remove max keystrokes
02960           LD      C,15        ;cursor off
02970           CALL    DSP
02980           POP     HL          ;remove ret adddress
02990           JP      START1      ;go for new label
03000 INKEY4    CP      16          ;is it ctrl-p ?
03010           JR      NZ,INKEY5   ;no - go on
03020           LD      C,15        ;cursor off
03030           CALL    DSP
03040           POP     BC          ;remove max keystrokes
03050           POP     HL          ;remove return address
03060           JP      PRINT0      ;goto print routine
03070 INKEY5    CP      17          ;is it ctrl-q ?
03080           JR      NZ,INKEY6   ;no - go on
03090           LD      C,15        ;cursor off
03100           CALL    DSP
03110           POP     BC          ;remove max keystrokes
03120           POP     HL          ;remove ret address
03130           JP      QUIT        ;go to quit routine
03140 INKEY6    CP      31          ;is it SHIFT CLEAR ?
03150           JR      C,INKEY1    ;smaller - do not allow
03160           JR      NZ,INKEY7   ;another chr - go on
03170           LD      C,15        ;cursor off
03180           CALL    DSP
03190           POP     BC          ;remove max keystrokes
03200           POP     HL          ;remove ret address
03210           JP      YOURS       ;goto the yours routine
03220 INKEY7    LD      D,A         ;keystroke ok
03230           POP     BC          ;restore max keystrokes
03240           LD      A,B         ;copy it to a
```

Lines 02470-02480 increments register L twice so, when we CALL the LOCATE routine in line 02490, the cursor is moved two spaces to the right. Line 02500 loads register C with the 'dash' character (or 'minus' character, whatever you prefer to call it), and line 02510 displays the character by CALLing the DSP routine. We have now displayed the first field ID: **1 -**

Continuing with line 02520, the original cursor position is POPped off the stack and into register HL. Since this position reflects the current line, and we wish to move to the next line, we simply increment the value in register H (line 2530). The cursor will then be positioned correctly when the loop brings the program back to the CALL LOCATE instruction in line 02450.

Now we restore the loop counter and the character, by POPping it off the stack and back into BC (line 02540). Register C is incremented in line 02550, thus giving us the next number to display when the loop brings the program back to the CALL DSP instruction in line 02460.

Finally, in line 02560, the DJNZ ELOOP3 instruction checks if register B has reached 0 (zero). If not, B is decremented, and program flow goes back to line 02430 and executes the loop again. When B reaches 0, we fall through to line 02570, where the routine RETurns to the caller.

In the last issue, I described the next subroutine (INKEY) as 'the heart of the program'. Indeed, it is just that. From the very first time it is accessed, it takes charge and determines the program flow from then on.

INKEY is first used from another subroutine, INPUT1, in lines 03360-03400. The purpose of INPUT1 is to set up the appropriate registers to get ready for the INKEY routine. Lines 03360-03379 loads HL with 0416H and then CALLs LOCATE, thus positioning the cursor at line 4, column 22. Register IX is loaded with the address of NABUF (this is the NAME BUFFER - the series of memory locations we will use to store the users input from the keyboard). Line 03390 (INPUT0) loads BC with 285FH. Once again, think of this as B=28H (40), and C=5FH (95). B (40) is the maximum number of characters we will allow the user to type, and C (95) is the character we will display to indicate the length of the input field.

Now that we are ready for the INKEY routine, we JP INKEY. Make sure you keep in mind that we did not CALL INKEY, we JumPed. This means that, at this moment, the top of the stack has the RETurn address to the CALLer of INPUT1, which is line 00870 (from the listing in last issue).

You got that, I hope!

Now we are in the INKEY routine, and the first thing we will do is to draw 40 of the underline characters to show the length of the user input field. To do that, we need to PUSH BC onto the stack, to preserve both the maximum characters allowed and the underline character (line 02590). The ILOOP1 (lines 02600-02610) loops 40 times to display 40 underline characters next to each other.

```
03250           CP      E           ;reached the max  ?
03260           JR      Z,INKEY8    ;yes - do not allow
03270           LD      C,D         ;no - keystroke to c
03280           CALL    DSP         ;and display it
03290           INC     L           ;move cursor right
03300           INC     E           ;chr count=chr count+1
03310           LD      (IX),C      ;store keystroker
03320           INC     IX          ;next buffer location
03330 INKEY8    PUSH    BC          ;save max keystrokes
03340           JP      INKEY1      ;go for next keystroke
03350 ;
03360 INPUT1    LD      HL,0416H    ;cursor @(4,22)
03370           CALL    LOCATE
03380           LD      IX,NABUF    ;point ix to buffer
03390 INPUT0    LD      BC,285FH    ;b=max chrs, c=chr
03400           JP      INKEY
03410 ;
03420 INPUT2    LD      HL,0516H    ;cursor @(5,22)
03430           CALL    LOCATE
03440           LD      IX,ADBUF    ;point ix to buffer
03450           JR      INPUT0
03460 ;
03470 INPUT3    LD      HL,0616H    ;cursor @(6,22)
03480           CALL    LOCATE
03490           LD      IX,CSBUF    ;point ix to buffer
03500           JR      INPUT0
03510 ;
03520 INPUT4    LD      HL,0716H    ;cursor @(7,22)
03530           CALL    LOCATE
03540           LD      IX,CNBUF    ;point ix to buffer
03550           JR      INPUT0
03560 ;
03570 INPUT5    LD      HL,0816H    ;cursor  @(8,22)
03580           CALL    LOCATE
03590           LD      IX,OTBUF    ;point ix to buffer
03600           JR      INPUT0
03610 ;
03620 KEY       PUSH    DE
03630           LD      A,1         ;@key
03640           RST     40
03650           POP     DE
03660           RET
03670 ;
03680 LOCATE    PUSH    DE
03690           PUSH    HL
03700           LD      B,3         ;b=function number
03710           LD      A,15        ;@vdctl
03720           RST     40
03730           POP     HL
03740           POP     DE
03750           RET
03760 ;
03770 PRINT     LD      A,14        ;@print
03780           RST     40
03790           RET
03800 ;
03810 PRT       LD      A,6         ;@prt
03820           RST     40
03830           RET
03840 ;
03850 PUTBUF    LD      A,(HL)      ;put chr in a
```

Line 02620 sets up register E as our character counter. As we haven't as yet typed anything, the counter is initialized as 0 (zero).

This portion of the routine is accessed only once each time we enter INKEY. The rest of the routine is accessed over and over, until particular keys are pressed. Keep in mind that we now have two unresolved values on the stack: the address of the original CALLer to INPUT1, and the value of BC.

INKEY1 begins the actual routine. In line 02630, CALL LOCATE positions the cursor according to the value of HL. Lines 02640-02650 turns the cursor on (as PRINT CHR$(14) in Basic).

Line 02660 is where everything happens. CALL KEY is a call to the subroutine which contains the @KEY SVC. This routine allows the user to type one character from the keyboard. This character is then stored in register A by the SVC.

O.K., we can go to work. We have a keypress, so now let's examine what user wants. Line 02670 checks if the key pressed is the <ENTER> key. We all know that <ENTER> is CHR$(13), right?

Line 02680 decides whether or not <ENTER> was pressed. If not, we jump to INKEY2 in line 02810. If it was pressed, on the other hand, the flow of the program falls through to line 02690. Here, character 13 (0DH) is stored at the address pointed to by IX (the current location in the storage buffer). The cursor is turned off (lines 02700-02710), and line 02720 POPs the maximum characters allowed back into register B (though the value of C is also restored, it is now of no importance to us).

The remaining portion of the input lenght indicator (the underline) needs to be erased, so line 02730 copies the maximum character allowed from B to register A. Line 02740 then subtracts the value in the character counter (register E) from register A. If the user has used the entire input field (typed 40 characters), the result of this subtraction would then be 0 (zero). We cannot allow the result to be 0, because the upcoming DJNZ loop will misinterpret and loop 256 times, rather than not loop at all. Thus, in line 02750, we take whatever value we got from the subtraction and add 1 to the result. This way the value will never be 0.

Now that we know how many of the underline characters remain on the line, we can loop to erase them. Line 02760 copies the amount of the remaining underline characters from A to register B. Line 02770 stores the erase character (space) in register C. Lines 02780-02790 then loops and erases the remaining underline characters, followed by a RET to the CALLer (line 02800).

This was nice and neat. The integrity of the stack was maintained.

We get to the INKEY2 routine (line 02810) if a character other than <ENTER> was pressed, so we test to see if it is CHR$(8) (LEFT-ARROW), indicating the user wishes to erase the previous character.

```
03860          CP      3           ;is it terminator ?
03870          JR      Z,PUTRET    ;yes - go terminate
03880          LD      (DE),A      ;no - store chr in buffer
03890          INC     DE          ;next buffer location
03900          INC     HL          ;next chr
03910          JR      PUTBUF      ;go do it again
03920 PUTRET   LD      A,13        ;cr in a
03930          LD      (DE),A      ;store cr in buffer
03940          RET                 ;return to caller
03950 ;
03960 TSTCHR   PUSH    HL          ;save buffer position
03970          LD      B,E         ;get chr count
03980 TLOOP    LD      A,(HL)      ;get chr
03990          CP      30H         ;is it "0" ?
04000          JR      C,TSTBAD    ;no - smaller - jump
04010          CP      3AH         ;is it larger than "9" ?
04020          JR      NC,TSTBAD   ;yes - larger - jump
04030          INC     HL          ;next chr - if any
04040          DJNZ    TLOOP       ;continue until done
04050          XOR     A           ;force z flag
04060          POP     HL          ;restore buffer location
04070          RET                 ;return - input good
04080 TSTBAD   OR      A           ;force nz flag
04090          POP     HL          ;restore buffer location
04100          RET                 ;return - input bad
04110 ;
04120 YOURS    LD      HL,0400H    ;cursor @(4,0)
04130          LD      E,63        ;e=number of chrs to
04140          CALL    ERASE       ;erase horizontally
04150          LD      HL,0416H    ;cursor @(4,22)
04160          CALL    LOCATE
04170          LD      HL,YNAME    ;display at cursor
04180          CALL    DSPLY
04190          LD      DE,NABUF    ;point de to buffer
04200          CALL    PUTBUF      ;copy to buffer
04210          LD      HL,0516H    ;cursor @(5,22)
04220          CALL    LOCATE
04230          LD      HL,YADDRS   ;display at cursor
04240          CALL    DSPLY
04250          LD      DE,ADBUF    ;point de to buffer
04260          CALL    PUTBUF      ;copy to buffer
04270          LD      HL,0616H    ;cursor @(6,22)
04280          CALL    LOCATE
04290          LD      HL,YCSZ     ;display at cursor
04300          CALL    DSPLY
04310          LD      DE,CSBUF    ;point de to buffer
04320          CALL    PUTBUF      ;copy to buffer
04330          LD      HL,0716H    ;cursor @(7,22)
04340          CALL    LOCATE
04350          LD      HL,YCNTRY   ;display at cursor
04360          CALL    DSPLY
04370          LD      DE,CNBUF    ;point de to buffer
04380          CALL    PUTBUF      ;copy to buffer
04390          LD      HL,0816H    ;cursor @(8,22)
04400          CALL    LOCATE
04410          LD      HL,YOTHER   ;display at cursor
04420          CALL    DSPLY
04430          LD      DE,OTBUF    ;point de to buffer
04440          CALL    PUTBUF
04450          JP      EDIT
```

If (LEFT-ARROW) was not pressed, the program jumps to INKEY3 in line 02930. If it was pressed, we continue with line 02830, where the character counter (register E) is copied to A. Line 02840 uses the instruction OR A to test if A is 0 (zero). If it is, we obviously cannot allow a backspace, so line 02850 sends the program back to INKEY1. If the counter is larger than 0, characters have been typed, and lines 02860-02870 moves the cursor left one position. Line 02880 decrements the character count, and lines 02890-02900 replaces the displayed character with the underline character. Line 02910 moves the buffer pointer (IX) back one position, and line 02930 sends the program flow back to INKEY1. Backspace and erase accomplished.

INKEY3 (line 02930) checks if <CTRL-N> was pressed. If so, the user wishes to start a new label. Line 02940 determines if the key was pressed. If not, we jump to INKEY4 in line 03000. If <CTRL-N> was pressed, line 02950 removes the top value on the stack and POPs it into register BC. It is of no importance what the value of BC is, we just want to get it off the stack. Lines 03020-03030 turns off the cursor. Line 02980 removes the RETurn address from the stack and POPs it into HL. Again, it is of no importance what that value is, we just want to remove it from the stack.

Having performed these two POPs, we have adjusted the stack to what it was BEFORE the original CALL to the subroutine. We can now JumP anywhere in our program, and line 02990 does just that. It JumPs to START1, which is routine to start a new label.

INKEY4 (line 03000) does basically the same thing as INKEY3. The difference is that it checks for <CTRL-P>. If that was not the key pressed, we go on to INKEY5 in line 03070. If <CTRL-P> was pressed, the user wishes to print the label. The cursor is turned off (lines 03020-03030), stack integrity is maintained (lines 03040-03050), and we go directly to the print routine by using the instruction JP PRINT0 (line 03060).

INKEY5 (line 03070) performs as INKEY3 & INKEY4. It checks for <CTRL-Q>. If not pressed, the program moves on to INKEY6 in line 03140. If pressed, the user wants to quit, so the cursor is turned off (lines 03090-03100), the integrity of the stack is maintained (lines 03110-03120), and we quit by JumPing directly to QUIT (line 03130).

INKEY6 checks if the user has pressed <SHIFT-CLEAR> to bring up his/her special label. First we check if the key pressed has a smaller value than <SHIFT-CLEAR>. If it does, we will not allow it, and line 03150 sends the program back to INKEY1. Line 03160 handles larger than, or equal to <SHIFT-CLEAR> If it is not equal to, we continue at INKEY7 in line 03220. If <SHIFT-CLEAR> was pressed, we turn off the cursor (lines 03170-03180), POP BC & HL to maintain the integrity of the stack (lines 03190-03200), and then JumP to the YOURS routine, which takes care of the special label.

INKEY7 (line 03220) is where we get to when a normal key is pressed. The first thing we do is copy the character to register D. Line 03230 restores the maximum keystrokes

```
04460 ;
04470 ;messages
04480 ;
04490 MSG1    DEFM  'TRSTimes  presents: '
04500         DEFB  143
04510         DEFB  244
04520         DEFB  245
04530         DEFB  246
04540         DEFB  3
04550 MSG2    DEFM  'TRSLABEL4'
04560         DEFB  3
04570 MSG3    DEFM  '(c) Lance Wolstrup'
04580         DEFB  3
04590 MSG4    DEFM  'a quickie mailing label program for
Model 4'
04600         DEFB  3
04610 NAME    DEFM  'Name:'
04620         DEFB  3
04630 ADDRS   DEFM  'Address:'
04640         DEFB  3
04650 CSZ     DEFM  'City, State & Zip:'
04660         DEFB  3
04670 CNTRY   DEFM  'Country:'
04680         DEFB  3
04690 OTHER   DEFM  'Other:'
04700         DEFB  3
04710 YNAME   DEFM  'TRSTimes magazine'
04720         DEFB  3
04730 YADDRS  DEFM  '20311 Sherman Way, Suite 221'
04740         DEFB  3
04750 YCSZ    DEFM  'Canoga Park, CA. 91306'
04760         DEFB  3
04770 YCNTRY  DEFM  'U.S.A.'
04780         DEFB  3
04790 YOTHER  DEFM  ' '
04800         DEFB  3
04810 EDMSG   DEFM  '1 - 5 to edit, ‹CTRL-N› for new
label,'
04820         DEFM  ' ‹CTRL-P› to print,'
04830         DEFM  ' or ‹CTRL-Q› to quit '
04840         DEFB  3
04850 HOWMNY  DEFM  'How many labels do you wish to
print (0 - 999) '
04860         DEFB  3
04870 ;
04880 ;buffers
04890 ;
04900 NABUF   DEFS  41
04910 ADBUF   DEFS  41
04920 CSBUF   DEFS  41
04930 CNBUF   DEFS  41
04940 OTBUF   DEFS  41
04950 EDBUF   DEFS  2
04960 NUMBUF  DEFS  4
04970 ;
04980         END   START
```

allowed in register B, which is then copied to register A (line (3240).

A check to see if the maximum number of characters (register E) have been typed is performed in line 03250). If we are at the maximum, we jump to INKEY8, not allowing the keypress. If not at the maximum, we copy the character from register D to register C (line 03270), and display it on the screen (line 03280). The cursor is moved one position to the right (INC L, in line 03290), and the character counter is incremented by one (line 03300). The character is copied to the buffer position pointed to by IX (line 03310), and the buffer position is moved up (line 03320).

INKEY8 (line 03330) saves the maximum keystrokes allowed (register C is now unimportant), and we go back to fetch another keypress by issuing the instruction JP INKEY1 (line 03340).

This was a fairly large routine, but do notice that it was broken down into much smaller routines, each handling a particular keypress. It makes it much easier.

INPUT1 (line 03360) was discussed earlier, suffice to say that the rest of the INPUT routines (2,3,4, and 5) perform the same job as INPUT1. The differences being the cursor location (register HL) and the buffer location (IX). Each routine ends up at INPUT0 (line 03390), where BC is PUSHed onto the stack and a JumP to INKEY is made (line 03400).

KEY (lines 3620-03660) is the @KEY SVC (get a character from the keyboard). Since register DE will be destroyed, we PUSH it onto the stack in line 03620, and POP it back after the SVC has executed (line 03650).

LOCATE (lines 03680-03750) is the @VDCTL SVC. This SVC has many uses. Function 3, which is loaded into the B register (line 03700), positions the cursor at the location specified by HL (H=row, L=column). We make sure that HL and DE return from the subroutine unchanged by PUSHing them onto the stack (lines 03680-03690), and POPping them back into the registers after the execution of the SVC (lines 03730-03740).

PRINT (lines 03770-03790) is the @PRINT SVC (sends string of characters pointed to by HL to the printer).

PRT (lines 03810-03830) is the @PRT SVC (sends character stored in register C to the printer).

PUTBUF (lines 03850-03940) is CALLed from the YOURS subroutine (lines 04120-04450 and is explained below). PUTBUF simply stores each field of your special label in the appropriate buffers. Before entering this routine, register HL will point to the text from message table, and register DE will point to the correct buffer. Line 03850 copies the character pointed to by HL into register A. Line 03860 checks if it the terminationg character (3). If so, we jump to PUTRET in line 03920). If not the terminator, then the character is stored in the buffer (line 03880). Line 03890 points DE to the next buffer location, and line 03900 points HL to the next character in the text. Line 03910 sends the program flow back to PUTBUF, continuing the copying of characters from the text to the buffer.

PUTRET (lines 03920-03940) is reached when register HL points to the terminating character (3). In line 03930 register A is the LoaDed with 13 (carriage return), which is then stored as the final character in the buffer (line 03930), followed by line 03940, which RETurns the routine to the caller.

TSTCHR (lines 03960-04100) is CALLed from PRINT1 (last issue). The purpose of the routine is to make sure that the prompt asking how many labels to print is answered correctly; that is, only the numbers 0-9 has been pressed.

Line 03960 PUSHes the address of NUMBUF (where the input is stored) onto the stack. Line 03970 copies the count of characters entered from register E (this value was obtained in the INKEY reitine) to register B so it can be used as a loop counter. Line 03980 (TLOOP) copies the character from NUMBUF (HL) to register A. Line 03990 compares the character to "0". If it is smaller (line 04000) then it is obviously not a number, and we jump to TSTBAD in line 04080. If not smaller than "0" we get to line 04010, where we compare the character to the ASCII value one larger than "9". If it is equal or larger than that (line 04020) it is not a number, and we jump to TSTBAD. Falling through to line 04030 means that register A holds a valid number, so we INC HL (line 04030) to point to the next character (if any). Line 04040 decrements the value of B and, as long as not 0 (zero), continues the loop, thus checking all characters entered. If any of the characters checked turn out to be outside of the range 30H-39H, the program flow is sent to TSTBAD.

We get to line 04050 only if the input was all numeric characters, so there we give the instruction: XOR A. It stores a zero in register A and, more importantly in this case, sets the Z flag. Line 04060 restores the pointer to the start of NUMBUF to HL, and line 04070 RETurns to the caller (with the Z flag set).

TSTBAD (line 04080) is reached whenever the TSTCHR routine encounters a non-numeric character in NUMBUF, so by giving the instruction OR A, we force the NZ flag status. Line 04090 restores the pointer to NUMBUF (actually the POP HL is done to maintain the integrity of the stack), and line 04100 RETurns to the caller (with the NZ flag set).

The YOURS subroutine (lines 04120-04450) is used from INKEY whenever (SHIFT-CLEAR) is pressed. The purpose is to erase whatever label is stored in the buffers as well as displayed on the screen, and instead store the special label in the buffers and display it on the screen. Line 04120 sets up HL with the cursor position. Line 04130 LoaDs register E with the number of characters to erase on the screen, and line 04140 CALLs the ERASE routine, which erases the portion of the screen needed, as well as the individual buffer; also it displays the field ID's (all this was explained earlier).

Lines 04150-04160 position the cursor @(4,22), and lines 04170-04180 displays the name portion of the special label. Lines 04190-4200 point DE to NABUF and, by CALLing PUTBUF, the name portion of the label will be stored there.

Lines 04210-04220 position the cursor on the next line, @(5,22). The address portion of the label is then displayed (lines 04230-04240), and the address is then stored in ADBUF (lines 04250-04260).

Lines 04270-04280 position the cursor @(6,22), where the City, State & Zip portion of the label is displayed (lines 04290-04300). Lines 04310-04320 store the City, State and Zip information in CSBUF.

Next, the cursor is positioned @(7,22) in lines 04330-04340, and the Country portion of the special label is displayed on the screen (lines 04350-04360). This information is then stored in CNBUF (lines 04350-04360).

Finally, the cursor is positioned @(8,22) in lines 04390-04400, and the OTHER portion of the label is displayed (lines 04410-04420). It is then stored in OTBUF (lines 04430-04440).

When all this information has been transfered (to the screen and to the buffers), line 4450 JumPs to the EDIT routine.

Lines 04490-4860 contain the different text for screen displays and prompts. Note that line 04710 should be changed to the name you wish for your special label. Line 04730 should be changed to the address for your label. Line 04750 should be changed to the city, state and zip for your label, and 04770 & 04790 should be changed to the country you wis and other information, respectively.

Lines 04900-04960 are the buffers we use to store the important information for later use. Note here that all, except EDBUF, are one character longer than the allowed input. This is done so, in case of maximum input by the user, we will still have room to store a terminating character. Without the terminator, our routines would not know where to quit.

Boy, this was a long one. I do hope that the concept of taking a program, such as TRSLABL4, breaking it down into managable routines, and explaining each in detail, has been of value to you. Let me know what you think.

Until next time....Keep on practicing.

A LITTLE HARD DISK PROBLEM part 3

By Roy Beck

Back to the grindstone. After much study of the disassembled code, I have found the answers to all of the questions posed initially. Let's review the results.

Patching the sign-on message with DU is trivial.

Head count happens to be already correct at four, but I wanted to know where it is stored in case I want to reapply this driver to still another drive at a future time.

Track count is the major difference. The driver was written for a 17 Meg drive with 480 tracks, and I am implementing a 5 Meg drive with only 153 tracks. I found several locations where track count is stored, and have now patched all of them.

I will digress a moment. In reviewing other applications of HD's, I have learned several things; first, the track to reduce write current and the track to introduce precompensation are typically about 1/2 way through the platter, second, the precise value appears not to be critical, third, this information is scarce, and finally, the reduction of write current is now usually done in hardware in the bubble (where it should be, anyhow). In my case, however, I am dealing with antique equipment, (as usual), and I need at least to be aware of these factors.

JBO in his System Programmer's Guide gives a different rule of thumb. My approach is to use the 1/2 factor unless I have specific information to the contrary. With my Hard Disk III I received some documentation, and to my surprise and pleasure, I found VR DATA has tabulated the above factors for some of the bubbles they offered in this box. For my Tandon TM-602S drives, the reduce write current (RWC) value is track 128 and the precomp (WPC) should begin at track 153. Since the tracks are numbered 0-152, the correct interpretation is that no precomp is required for these particular bubbles. As a check on this conclusion, I examined the driver Roy Soltoff wrote for this same Hard Disk III.

Here I must further digress. The nameplate on the Hard Disk III shows the box originally contained one Tandon TM-501S bubble. That bubble had only 1 platter, two heads, and 306 tracks. In addition it showed a "step option" of 4. Since the bubble expected by Soltoff is not the same one I am dealing with, I have had to interpret and modify my data. My TM-602S bubble has "ramped seek", and therefore I can tell the driver to use the fastest step rate of which it is capable, as the ramped seek function will control the actual stepping rate. Evidently the older TM-501S bubble did not have ramped seek and could not accept the fast step rate. The Tandon manuals show it has a step rate of 3 milliseconds, with no mention of ramped seek. Evidently, Soltoff tried various values to see what works reliably. For whatever reason, he settled on step option 4 as his default, which is also the value on the nameplate of the box. VR DATA supplied DOSPLUS with its own drivers, and advised the owner to use the step rate value marked on the nameplate.

When I implemented TRSDOS V 6.3 on drive 0, I used step rate 0, in the belief the TM-602S would buy this and operate in its ramped seek mode. Since that was successful, I am using the same step rate in JBO's CP/M driver.

I mentioned the directory track above. My reason for mentioning it at all Is that TRSDOS usually locates the directory at the midpoint of head travel in order to minimize access time. Accordingly, the 153 track drives normally have the directory on track 76. But because of the structure of CP/M, it will be on the first non-system track, track #2 in my case.

I believe there in no diagnostic track in the CP/M implementation, but I have not proved that yet. I believe JBO skipped a diagnostic track on the basis that the two outer tracks are reserved for the CP/M system, and if diagnostics need to be performed, those tracks are available, and the system can be restored, if necessary. No proof of this, just my theory.

The drive partitioning tables in JBO's driver have to be tailored to fit the drive. Changing the maximum track number from 480 to 153 is no problem. But the associated constants in the Drive Parameter Block (DPB) are another matter. Fortunately, Dave Cortesi explains these adequately in his manual, "Inside CP/M".

Part of my task is to assign Allocation Block sizes to the various partitions possible with 1, 2, 3 or 4 logical partitions. In the case of the 17 Meg drive, JBO assigned the sizes as follows:

For one logical partition of 17 Meg, he used a 16K allocation block. This is large, but necessary in order not to run out of directory space.

For two logical partitions, he used 8K blocks. This arrangement pairs the heads, with two heads making up each partition.

3 partitions is a little strange. He assigns two heads (1/2 of the drive) to the first partition, and uses 8K blocks. The other two partitions are assigned one head each, with 4K blocks. CP/M's structure allows this, no problem.

Finally, with 4 partitions, each partitlon gets one head and uses 4K blocks.

Since my drives are only 1/3 the size of his, I wanted to reduce the block size. However, to reduce the amount of time I have to invest in this effort, I elected to keep hls block size. Purely an expedient decision, I acknowledge. Maybe at

a later date I will go back to this area, but not for now. I must start my income tax effort SOON, and so I must wrap up this one right away.

The constants to go into the DPB are the values considered in the foregoing couple of paragraphs, so I ended up only changing the maximum track count from 480 to 153.

Does it all work? Well.....

The answer, so far, is a qualified yes. When I had made all the patches I thought necessary, I backed up the other bubble (you never know), backed up my patched (and renamed) driver, and gave it the smoke test.

Yes, the correct bubble activity light blinked.

Yes, the format function stepped through the correct number of tracks.

Yes, CP/M appeared to be running from my new drive A on the hard drive.

Would it boot up correctly? Yes, it will properly reboot from the floppy.

Next step was to PIP files from the floppy to the HD. I gave the command:

B:PIP A:=B:*.*

The drives whirred, the lights blinked, and the DOS came back with a diagnostic to the effect that either my HD had no more room, or my directory was full or my directory didn't exist, or all of the above!

What was wrong? Thinking about the problem, I remembered a quirk of the Xebec HDC. Unless told otherwise, the HDC will entirely format a bubble with 6C. But CP/M insists that unused directory directory entries must begin with E5. I also remembered the code did have a little section to cause the HDC to use E5 instead of 6C. Furthermore, it had a test later to verify the E5 was used. So what was in my directory?

Loading DU, I began exploring. Right away, I found the directory sectors were full of 6C. Using a DU function, I rewrote block 0 (the first half of the directory) with E5. Try PIP again. Hooray! It works. So why did the format function (or something) fill the directory with 6C instead of E5? I dunno. I will study the problem, but at the moment I have no idea. Note that the test to verify E5 did not report a problem. So, either the trouble occurred after that test, or the test was bypassed!

After some more digging, I learned the apparent cause of the trouble. In the string of 6 bytes which must be issued to the HDC for each command, the last byte contains several pieces of information, including the step rate and a bit which must be set in order for the format command to use the contents of the internal sector buffer. Obviously all the commands except FORMAT ignore this bit, and since I had not set it, the HDC used its default 6C instead of E5 when formatting the bubble.

As an aside, I had noticed that JBO had set this bit, but the Xebec "manual" I was following made no mention of it. Soltoff's TRS version did not set the bit, so I didn't either, originally. It was not until this trouble occurred that I started digging in earnest on this point.

I have several different sets of documentation on the SASI/SCSI/Xebec HDC, with some inconsistencies between them. I finally found the FORMAT situation described in one of them with enough information to make clear what was going on. I set the missing bit, FORMATted the bubble again, and all seems to be OK now. After the fact, I found confirmation of this in Cortesi's book. This also explains a small detail I found in one of the various TRSDOS HD Installation manuals. That said to install CP/M first, then follow with TRSDOS. That probably is referring to this same need to format with E5 to keep CP/M happy.

The actual changes in the driver, when all is said and done, amounted to about 60 bytes. In retrospect, I decided I must not alter the head count, as the structure of the driver would have to change fundamentally. However, the driver is quite usable as it is. Because of the way JBO organized the driver, one can enable 1, 2, 3 or 4 partitions, each of which partitions the drive in a different way. In my case, I opted for 4 partitions, with only one activated. This had the effect of assigning only one head to CP/M, (Drive A: on head #4), and left the other 3 heads unused. The drive is 1/4 of 5 Megs, and even with CP/M on it, I have plenty of space available for my meanderings.

Finally, I think I have reached the end of the trail, although the pot of gold there is rather meager. The CP/M boot disk boots correctly, I have 1.25 Megs of HD for drive A: and now I can begin to study dBase II, which was after all the reason for this little project.

The command string I have in my SUBMIT file is as follows: **HDVRD05A H1=***A F=BCDE**

This instructs the driver to allocate for 4 partitions, but only assigns the last one (head #4 of 4) to A:, and to assign 4 floppies to B, C, D and E, respectively. I then installed LDOS on the 3 remaining heads. just to facilitate operations, I also assigned one head on the other bubble to LDOS. This gives me one partition in common with TRSDOS, which I assigned as drive :2 on both LDOS and TRSDOS. Because both of those DOSes use the same directory structure, this allows me to shift files from one DOS to the other by simply saving them on drive :2 of either DOS and accessing them on the other DOS!

I have now written 11 articles on CP/M over the past two years. I will freely admit it is a bit of an ego trip to see one's work in print, but it is disappointing to receive so little feedback from you readers. Are you reading them? Are you interested in them? Are they of value to you? Is it a lack of interest in CP/M? If that is the case, should I write on some other topics? I would surely appreciate a few comments.

-Roy-

THE SWAP MEET

WANTED: Any literature, such as owner's or tech manuals for the Model III in Spanish. Basic manuals or anything of likely value to beginners. Will buy or pay to duplicate.
Call Harold May. (312) 325-1910 collect.

WANTED: Any software/hardware to allow the Model III or IV to communicate with H.P. xy recorders, analog or digital type.
John Greenland. Box 171,
Kelligrews, New Foundland. A0A 2T0 Canada

FOR SALE: TRS-80 SOFTWARE, Models 1/3/4/4P/4D.
Many useful programs. Economical prices.
Send $3 for listing.
Practical Programs, 1104 Aspen Drive.
Toms River, NJ. 08753 (201) 349-6070

FOR SALE: One Radio Shack RS-232C Serial Interface with cables for Model III. With shipping, $15
Howard W. Mueller, Box 17, Pocahontas, MO. 63779

FOR SALE: Printer Buffer, Centronics Port compatible (IBM PC + others), 64 KBytes (25 pages), Reset/Bypass/Copy buttons, 8 LED indicators (status + memory fullness), 5x7x2 inch metal case, 2 pounds, AC/DC with Power supply, builtin Selfcheck, 1 year guarantee, includes shipping - $119. Call/write:
Practical Programs, 1104 Aspen Drive.
Toms River, NJ. 08753 (201) 349-6070

WANTED: I am looking for back issues of the Radio Shack Computer Catalog to complete my collection. I specifically need the following issues: RSC-1, 3, 13, 18E, 19E, 20E, 21E, and Software Buyers Guide 1st Edition. Any assistance would be greatly appreciated.
Roy Beck. 2153 Cedarhurst Dr.
Los Angeles, CA. 90027

WANTED: To complete collection, I am looking for SOFTSIDE Nov 80, Dec 80, and Jan 81. Also looking for any and all issues of PROG 80.
Lance Wolstrup. 20311 Sherman Way. Suite 221
Canoga Park, CA. 91306

WANTED: LISP, PROLOG, APL, ADA, C+ +, etc. for Model III/4. Also RS Service Repair Manuals for Mod III/4, Forth Robot Arm for Mod III/4. Wish to contact folks who are in to Robotics, AI, Neural Networks, Nanotechnology.
R. Yves Breton. C.P. 95, Stn. Place D'Armes
Montreal, Quebec, Canada H2Y 3E9

FOR SALE: One Grafyx Solution Hi-Resolution Board for Model III. Includes DRAW program and GBASIC 2.0 Enhancement Program. With Manuals and Disks. $65.00 with shipping.
Howard W. Mueller, Box 17, Pocahontas, MO. 63779
(314) 833-6922

FOR SALE: PD GOOD GAMES FOR MODEL I/III.
GAMEDISK#1: amazin/bas (maze), blazer/cmd (arcade), breakout/cmd (break down the walls), centipede/cmd (arcade), elect/bas (a simulation of the 1980 election), madhouse/bas (adventure), othello/cmd (board), poker/bas (almost better than going to las vegas - well, cheaper!!), solitr/bas (great solitaire card game). towers/cmd (puzzle game).
GAMEDISK#2: crams2/cmd (chase), falien/cmd (arcade), frankadv/bas (adventure), iceworld/bas (adventure game), minigolf/bas (putt-putt on the trs-80), pingpong/cmd (1 or 2 player arcade game), reactor/bas (simulation), solitr2/bas (another good solitaire card game), stars/cmd (2 player race game), trak/cmd (maze game).
GAMEDISK#3: ashka/cmd (d&d), asteroid/cmd (arcade), crazy8/bas (card game), french/cmd (space invaders in french), hexapawn (board game), hobbit/bas (adventure game), memalpha (adventure game), pyramid/bas (good solitaire card game), rescue/bas (arcade game), swarm/cmd (arcade game).

Price per disk: $5.00 (U.S.)
or get all 3 disks for $12.00 (U.S.)
TRSTimes - PD GAMES. 20311 Sherman Way. Suite 221
Canoga Park, CA. 91306

TRSTimes on DISK #5
Issue #5 of TRSTimes on DISK is now available, featuring all the programs from the January, March and May 1990 issues.

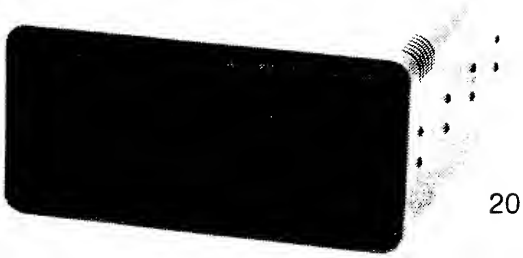
U.S. & Canada: $5.00 (U.S.)
Other countries:$7.00 (U.S.)

Send check or money order to:
TRSTimes on DISK
20311 Sherman Way, Suite 221
Canoga Park, CA. 91306. U.S.A.

TRSTimes on DISK #1, 2, 3 & 4
are still available at the above prices

PERIPHERALS SALE!

Model 3/4 Hard Disk Drives



- Lowest Prices Ever
- Brand New Units
- FCC Class B Certified
- Complete with Cables
- Complete with Software
- Money Back Guarantee
- High Performance
- Reliable
- Thousands in Use

As Low As **$289** 5MB. 80ms

20MB, 65ms.. only **$449** 40MB, 40ms (28ms optional).. **$559**

Faster drives available at extra cost.
Add $22 packing and shipping in a custom made foam carton.
Brushed stainless steel case available. Add $20.

Software and cables included.

Aerocomp leads the way with lower prices for our loyal TRS-80 friends once again. Aerocomp drives are a TRS-80 standard and are available in three sizes. These are not uncertified kits, but new and complete units ready to run. All models include **brand new Seagate drives,** not some used drive or one that has been refurbished or from a second rate manufacturer's boneyard (Tandon, Miniscribe, etc.)

These external hard drives are FCC Class B Certified as required by law. Aerocomp hard drives are an established product and have survived the test of time. Thousands of satisfied Aerocomp hard drive customers have proven these products a solid value for their owners. A secondary hard drive can be added at any time you desire. Larger drives can be installed in your original case thereby protecting your investment. The hard drive itself can even be transferred to an MS-DOS compatible computer if that is in your future.

Our units are complete with a 6' interface cable and the TRSDOS, LDOS or CP/M software driver of your choice at no additional cost.

Aerocomp provides all the little things that are so important for a long, **trouble-free** life: continuous-duty switching power supplies, filtered forced-air ventilation, effective EMI filtration, solid steel construction, five front panel indicator lights (Power-Ready-Read-Write-Select), built-in diagnostics, and gold plated connectors.

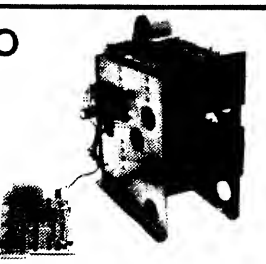
Probably the most important thing of all is our 30 day money back guarantee. If, for any reason, you are not satisfied with the drive, we'll refund the entire purchase price, less the shipping charges. The warranty is for one full year and includes all parts **and** labor.

Save Now On Our LOW COST FLOPPYS

Aerocomp has been supplying quality disk drives at low prices since 1980. All drives are half-height and are new—not factory blems, seconds, close-outs or a defunct manufacturer's surplus (MPI, Qume, Tandon, etc.) We offer just about every combination of internal and external floppy configuration plus the proper cables to connect everything together. We appreciate your business and will do our very best to support you. If, for any reason, you aren't happy with your hardware selection, we'll cheerfully refund the entire purchase price, less shipping. Order yours today! All items (except software) have a one year parts and labor warranty.

BARE DRIVES	
360K 5.25" TEAC 55B bare drive 40tk	$79
720K 5.25" TEAC 55F bare drive 80tk	109
1.2M 5.25" TEAC 55G bare drive 80tk	85
360K 3.50" TEAC 35B bare drive 40tk	59
720K 3.50" TEAC 35F bare drive 80tk	69
1.4M 3.50" TEAC 35H bare drive 80tk	69
DRIVE-POWER SUPPLY COMBINATIONS (Includes gold plated extender)	
1-TEAC 35B 360K dual enclosure	$129
1-TEAC 35F 720K dual enclosure	169
1-TEAC 55B 360K dual enclosure	149
1-TEAC 55F 720K dual enclosure	$149
Add $10 for brushed stainless steel cover.	
CABLES - CASES - DOS	
IBM ext floppy cable (drives C/D)	$39
TRS-80 2-drive floppy cable	24
TRS-80 4-drive floppy cable	34
6' floppy ext cable gold contacts	12
3½" case, power supply w/o ext	49
5¼" case, power supply w/o ext	59
TRS-80 Model 4 CP/M (Monte ver.)	69
Add $10 for brushed stainless steel cover.	
Add $4 shipping for singles, $6 for duals.	

ADD DISK DRIVES TO YOUR MODEL 3/4!



New Low Price! Complete System Less Drives, DOS

Now Only **$99** Save!

Convert your cassette Model 3 or 4 to fast disk operation with **one of our** easy-to-install kits. Complete instructions are provided. All you **need is a** screwdriver and a pair of pliers. Our own advanced controller, **100%** compatible with the original, plated steel mounting towers **with RFI shield** and all cables and hardware included. Select your drives **from the other** column and call us, toll-free, to place your order. If, for **any reason, you** don't like the kit, we'll refund the entire purchase price, **less shipping.** Order yours today!

Disk Controller only $49

RS-232 Board complete $49

Add $5 shipping. One year parts and labor warranty.

DOUBLE DENSITY CONTROLLER

Now Only **$49** Add $4 Shipping

80% more disk capacity is what you get when you add our DDC to your TRS-80 Model 1. This controller has withstood the test of time. All the others are gone, yet the Aerocomp DDC endures. Why? Because it has proven itself as the only way to achieve reliable floppy disk operation on the Model 1. Requires the Radio Shack Expansion Interface and software driver. All DOS (except TRSDOS) have the necessary double density driver. If, for any reason, you don't like the DDC, we'll refund the entire purchase price, less shipping. Order yours today! One year parts and labor warranty.

OUT THEY GO!

ENJOY INCREDIBLE SAVINGS NOW ON SOFTWARE, BOOKS AND MANUALS...WHILE THEY LAST!

CP/M® SOFTWARE	
Twist & Shout	$6
CP/M BOOKS & MANUALS	
Inside CP/M, by Cortesi (book)	$2
CP/M System Prog. Manual, Model 4	5
Monte's Mail (newsltr), Vol 1 #1, Vol 2 #1	
Specify Volume ..... Each Volume	1
TRS-80 SOFTWARE	
BASIC Faster & Better Library Disk	$2
BASIC Faster & Better Demo Disk	$2
BASIC I/O Demonstration Disk	2
TRSDOS 6.2 Utilities with Manual	9
Electric Pencil Word Processor	9
TRS-80 BOOKS	
Games & Graphics for the TRS-80	$1
Inside Level II	1
Tandon 848-1 Service Manual	5

Add $2 shipping per order for books

AEROCOMP

2544 West Commerce St Dallas, Texas 75212

SERVICE: 214-638-8886
TELEX 882761

INFORMATION
214-637-5400

FAX 214-634-8303

MasterCard VISA TeleCheck

"SERVING YOU SINCE 1980"

ORDER TOLL FREE!

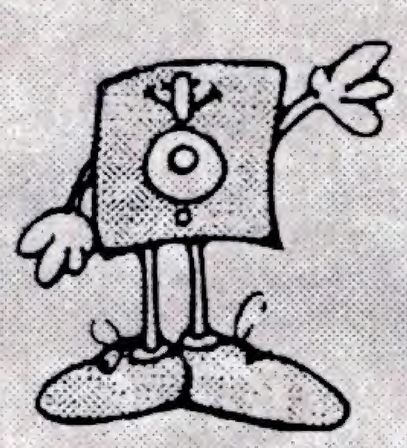
M-F 9-7; Sat 10-3

1-800-527-0347 AD Y1

Have your American Express, MasterCard or Visa ready. We will not charge your card until the day we ship your order. Mail orders are welcome. Money orders are accepted as well as your company and personal checks as long as they are bank printed and have your address and telephone number. We will ship surface COD with no deposit on most items, but all COD's raquire cash or a Cashier's Check on delivery. Texas residents add State Salas Tax. No tax collected on out of state shipments. There is a one year warranty (unless otherwise stated) on all hardware items against defects in materials or workmanship. Your satisfaction is guaranteed on hardware products. If you are not satisfied, for any reason, call us within 30 days of receipt and we will cheerfully refund your money (less shipping). All original materials must be intact and undamaged, as well as the original container. This offer does not apply to software. Defective software will be replaced. No other software warranty applies. Prices and specifications are subject to change without notice. Any returns must have our authorized RMA number on the label to be accepted.

NEW MODEL 4 SOFTWARE - NEW MODEL 4 SOFTWARE - NEW MODEL 4 SOFTWARE

TIRED OF SLOPPY DISK LABELS?
TIRED OF NOT KNOWING WHAT'S ON YOUR DISK?
YOU NEED 'DL'



- 'DL' will automatically read your TRSDOS6/LDOS compatible disk and then print a neat label, listing the visible files (maximum 16).
- You may use the 'change' feature to select (or reject) the filenames to print. You may even change the diskname and diskdate.
- 'DL' is written in 100% pure Z-80 machine code for efficiency and speed.

'DL' is available for TRS-80 Model 4/4P/4D
using TRSDOS 6.2/LS-DOS 6.3.0 & 6.3.1.
with either an Epson compatible or DMP series printer.

'DL' for Model 4 only $9.95

TRSTimes magazine - Dept. 'DL'
20311 Sherman Way, Suite 221
Canoga Park, CA. 91306

STORAGE POWER

Your SOURCE for Models I, III, IV's

HARD DISK DRIVES

We sell complete hard drive units. They may cost a little more. However, we only use quality components such as Western Digital controllers (not some out of production parts), our own high speed host adapter, 60 watt power supply, room for a second hard drive or HH floppy, and quiet, time proven quality drives. Tandon (made by W.D.) Miniscribe and others, Seagate avail upon request. Hard disk units can be changed over to MS-DOS if desired. All Hard Drive units come complete with cables and driver of your choice, (LDOS Mod I/III, TRSDOS 6.x, LS-DOS 6.x, MULTIDOS $10.00 Xtra)

10 Meg...$ 425.00 15 Meg...$ 495.00
20 Meg...$ 545.00 30 Meg & up $Call

Bare hard drive bubbles avail.CALL BBS
Storage Power HD host adapter...$ 59.95

HARD DISK DRIVERS:

We've been using & selling Powersoft drivers (the Best) for our drives and carry them for other brands including R/S.
Partition your HD by head or cylinder.
•Mod I/III LDOS.....$ 14.95
•Mod IV TRSDOS 6.x, LS-DOS 6.x
Includes HD boot for 4p.....$ 19.95
•Both for .....$ 29.95
MULTIDOS Hard Disk drivers.....$ 39.95

DISKETTES w/sleeves & labels

	5.25"	3.5"
Pkg of 10	$ 4.25	$ 11.95
Pkg of 25	$ 9.95	$ 25.95
100 5.25" Disk storage w/lock		$ 11.95
70 5.25" Disk storage w/lock		$ 9.95
40 3.5" Disk storage w/lock		$ 8.95
80 3.5" Disk storage w/lock		$ 12.95

TIMECLOCK Model IV's

- Automatic DATE and TIME when booting.
- Connects to and extends 50 pin buss.
- Lithium Battery backup
- Addressable from basic.
- Free standing or attaches to Computer.

Introductory price $ 39.95

MISCELLANEOUS

Power Supplys 65w Aztec.....$ 34.95
60w replacement for R/S 38w.....$ 59.95
CRT Tube green/amber.....$ 79.95
Mod I Double Density Board.....$ 89.95
Printer cables 6ft $ 14.95/ 12ft $ 19.95
34 pin edgecard cable connector....$ 1.25
Connectors, cable or custom cables $ CAll

We can supply most of the parts (new & used) that you will need in repairing & upgrading Mod I, III or IV's. Call or write for availability & price.

Call our BBS for SPECIALS and other products.
(714) 952-8666 8-N-1
STORAGE POWER
10391 Oakhaven Dr.
Stanton, Ca. 90680
(714) 952-2700
9:00 am - 8:00 pm PST
All C.O.D orders are cash only. Prices are plus shipping and subject to change and availability. Calif orders require 6.25% sales tax.

III/IV INTERNAL DISK DRIVE KITS

Complete with controller, drive stands, power supply, cables. Add Drives & Dos.
2 FH Drives $149.95..4 HH Drives $159.95

FDC controller only	$. 89.95
Internal 20 pin flat ribbon cable	$ 4.95
Int. disk drive cable non G/A	$ 9.95
Int. disk drive cable G/A	$ 12.95
Disk drive cable 2 drives 3ft	$ 9.95
For pin selected cables add	$ 3.00
Metal drive stands	$ 29.95

EXTERNAL DISK DRIVES

Complete w/case, power supply, Cables.

2 40 track HH DS DD	$ 229.95
2 80 track HH DS DD	$ 249.95
2 3.5" 80 track	$ 269.95
1 80 track FH DS DD	$ 119.95

BARE DRIVES

40 track DS DD FH refurb 360k	$ 64.95
Replacement for SS Mod III & IV	
40 track DS DD HH..360k	$ 79.95
80 track DS DD HH..720k	$ 89.95
80 track DS DD FH..720k	$ 49.95
3.5" 80 track..720k	$ 99.95

DRIVE CASES W/Power supply

Hard Disk 1 FH or 2 HH w/fan	$ 99.95
Floppy 1 FH or 2 HH	$ 59.95

MOD IV MEMORY SETS

Caution some people do not specify new versus pulls.

8 4164-200ns new	$ 14.95/Pulls $ 9.95
8 4164-150ns new	$ 19.95/Pulls $ 14.95
Pal chip for non Gate/array	$ 10.95

MOD IV SPEEDUP KITS

Non Gate array (5.1Mhz)	$ 34.95
Gate array (6.3Mhz)	$ 34.95

ATTENTION TRSDOS 1.3. USERS!

ANNOUNCING "SYSTEM 1.5.", THE MOST COMPREHENSIVE 1.3. UPGRADE EVER OFFERED!

MORE SPEED!! MORE POWER!! MORE PUNCH!!

While maintaining 100% compatibility to TRSDOS 1.3., this DOS upgrade advances TRSDOS 1.3. into the 90's!
SYSTEM 1.5. supports 16k-32k bank data storage and 4MGHZ clock speed (4/4P/4D).

DOUBLE SIDED DRIVES ARE NOW 100% UTILIZED! (all models).

CONFIG = Y/N	CREATES CONFIG BOOT UP FILE	DATE = Y/N	DATE BOOT UP PROMPT ON or OFF
TIME = Y/N	TIME BOOT UP PROMPT ON or OFF	CURSOR = 'XX'	DEFINE BOOT UP CURSOR CHAR
BLINK = Y/N	SET CURSOR BOOT UP DEFAULT	CAPS = Y/N	SET KEY CAPS BOOT UP DEFAULT
LINE = 'XX'	SET *PR LINES BOOT UP DEFAULT	WP = d.Y/N (WP)	WRITE PROTECT ANY or ALL DRIVES
ALIVE = Y/N	GRAPHIC MONITOR ON or OFF	TRACE = Y/N	TURN SP MONITOR ON or OFF
TRON = Y/N	ADD an IMPROVED TRON	MEMORY = Y/N	BASIC FREE MEMORY DISPLAY MONITOR
TYPE = B/H/Y/N	HIGH/BANK TYPE AHEAD ON or OFF	FAST	4 MGHZ SPEED (MODEL 4'S)
SLOW	2 MGHZ SPEED (MODEL III'S)	BASIC2	ENTER ROM BASIC (NON-DISK)
CPY (parm,parm)	COPY/LIST/CAT LDOS TYPE DISKS	SYSRES = H/B/'XX'	MOVE/SYS OVERLAY(s) TO HI/BANK MEM
SYSRES = Y/N	DISABLE/ENABLE SYSRES OPTION	MACRO	DEFINE ANY KEY TO MACRO
SPOOL = H/B.SIZE	SPOOL is HIGH or BANK MEMORY	SPOOL = D.SIZE = 'XX'	LINK MEM SPOOLING TO DISK FILE
SPOOL = N	TEMPORARILY DISABLE SPOOLER	SPOOL = Y	REACTIVATE DISABLED SPOOLER
SPOOL = RESET	RESET (NIL) SPOOL BUFFER	SPOOL = OPEN	OPENS, REACTIVATES DISK SPOOLING
SPOOL = CLOSE	CLOSES SPOOL DISK FILE	FILTER *PR.ADLF = Y/N	ADD LINE FEEDS BEFORE PRINTING 0DH
FILTER *PR.IGLF	IGNORES 'EXTRA' LINE FEEDS	FILTER *PR.HARD = Y/N	SEND 0CH to PRINTER (FASTEST TOF)
FILTER *PR.FILTER	ADDS 256 BYTE PRINTER FILTER	FILTER *PR.ORIG	TRANSLATE PRINTER BYTE TO CHNG
FILTER *PR.FIND	TRANSLATE PRINTER BYTE TO CHNG	FILTER *PR.RESET	RESET PRINTER FILTER TABLE
FILTER *PR.LINES	DEFINE NUMBER LINES PER PAGE	FILTER *PR.WIDTH	DEFINE PRINTER LINE WIDTH
FILTER *PR.TMARG	ADDS TOP MARGIN to PRINTOUTS	FILTER *PR.BMARG	ADDS BOTTOM MARGIN to PRINTOUT
FILTER *PR.PAGE	NUMBER PAGES, SET PAGE NUMBER	FILTER *PR.ROUTE	SETS PRINTER ROUTING ON or OFF
FILTER *PR.TOF	MOVES PAPER TO TOP OF FORM	FILTER *PR.NEWPG	SET DCB LINE COUNT TO 1
FILTER *KI.ECHO	ECHO KEYS to the PRINTER	FILTER *KI.MACRO	TURN MACRO KEYS ON or OFF
ATTRIB:d.PASSWORD	CHANGE MASTER PASSWORD	DEVICE	DISPLAYS CURRENT CONFIG INFO

All parms above are installed using the new LIBRARY command SYSTEM (parm,parm). Other new LIB options include DBSIDE (enables double sided drive by treating the "other side" as a new independent drive, drives 0-7 supported) and SWAP (swap drive code table #s). Dump (CONFIG) all current high and/or bank memory data/routines and other current config to a disk data file. If your type ahead is active, you can (optional) store text in the type buffer, which is saved. During a boot, the config file is loaded back into high/bank memory and interrupts are recognized. After executing any active auto command, any stored type ahead data will be output. FANTASTIC! Convert your QWERTY keyboard to a DVORAK! Route printer output to the screen or your RS-232. Macro any key,even F1, F2 or F3. Load *01-*15 overlay(s) into high/bank memory for a memory only DOS! Enter data faster with the 256 byte type ahead option. Run 4MGHZ error free as clock, disk I/O routines are properly corrected! Spool printing to high/bank memory. Link spooling to disk (spooling updates DCB upon entering storage). Install up to 4 different debugging monitors. Print MS-DOS text files, ignoring those unwanted line feeds. Copy, Lprint, List or CATalog DOSPLUS, LS-DOS, LDOS or TRSDOS 6.x.x. files and disks. Add top/bottom margins and/or page numbers to your hard copy. Rename/Redate disks. Use special printer codes eg: LPRINT CHR$(3); toggles printer output to the ROUTE device. Special keyboard codes add even more versatility. This upgrade improves date file stamping MM/DD/YY instead of just MM/YY. Adds optional verify on/off formatting, enables users to examine *01-*15, DIR, and BOOT sectors using DEBUG, and corrects all known TRSDOS 1.3. DOS errors. Upgrade includes LIBDVR, a /CMD driver that enables LIBRARY commands, such as DIR, COPY, DEBUG, FREE, PURGE, or even small /CMD programs to be used within a running Basic program, without variable or data loss.

By special arrangement with GRL Software,
SYSTEM 1.5. is now distributed exclusively by TRSTimes magazine.

ORDER YOUR COPY TODAY!

Send $39.95 (U.S. funds) to:

TRSTimes - SYSTEM 1.5.
5721 Topanga Canyon Blvd., Suite 4
Woodland Hills, CA. 91367